Stellar Occultation Reduction Analysis Documentation

Release 0.3

SORA Team

Jan 31, 2023

DETAILS:

(Legacy documentation for SORA v0.2.1 is available here.)

CHAPTER

ONE

OVERVIEW

1.1 Stellar Occultations

A stellar occultation occurs when a solar system object passes in front of a star for an observer on Earth, and its shadow causes a temporary drop in the observed flux of the star. This technique allows the determination of sizes and shapes with kilometre precision and to obtain characteristics of the object, such as its albedo, the presence of an atmosphere, rings, jets, or other structures around the body (Sicardy et al. , , Braga-Ribas et al. , , , Dias-Oliveira et al., , Benedetti-Rossi et al., , , Ortiz et al., , , , Leiva et al., , Bérard et al., , Morgado et al., , Gomes-Júnior et al., , Souami et al., , Santos-Sanz et al.,) or even the detection of topographic features (Dias-Oliveira et al.).

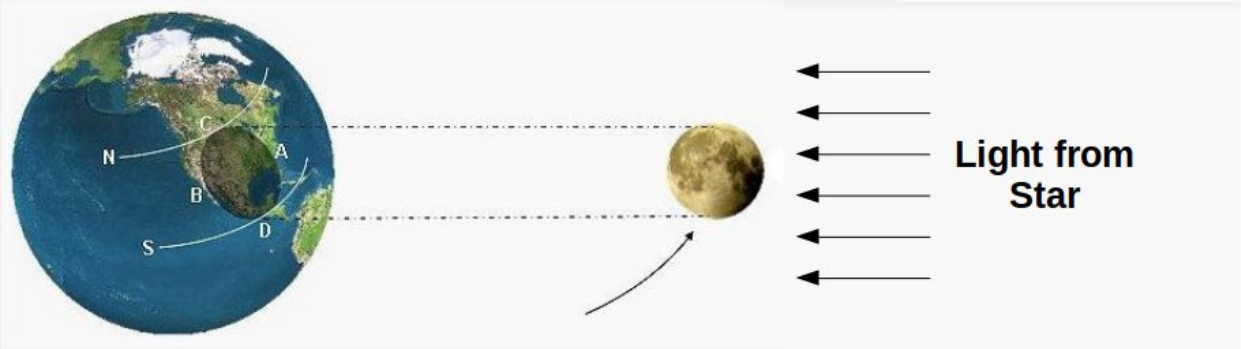


Figure 1: Illustration of a stellar occultation geometry. Original source: IOTA, editted by G. Benedetti-Rossi.

1.2 The SORA library

SORA is a Python-based, object-oriented library for optimal analysis of stellar occultation data. The user can use this library to build pipelines to analyse their stellar occultation's data. It includes processes starting on the prediction of such events to the resulting size, shape and position of the Solar System object. The main modules available at version 0.2 are: **star**, **body**, **observer**, **lightCurve** and **occultation**. It is important to note that new modules and other improvements and implementations can be available in future versions.

A stellar occultation is defined by the occulting body (**Body**), the occulted star (**Star**), and the time of the occultation. On the other hand, each observational station (**Observer**) will be associated with their light curve (**LightCurve**). SORA has tasks that allow the user to determine the immersion and emersion times and project them to the tangent sky plane, using the information within the **Observer**, **Body** and **Star** Objects. That projection will lead to chords that will be used to obtain the object's apparent size, shape and position at the moment of the occultation. Automatic processes were developed for optimising the reduction of typical events. However, users have full control over the parameters and methods and can make changes in every step of the processes.

The Classes developed within SORA were built as an integrated system, and they are controlled by the **Occultation** Class. Combining these Classes and the functions therein, the user can perform the complete data reduction, pre- and post-occultation. Jupyter-Notebooks with the features of each Class can be found in the example folder.

1.3 Acknowledgements

The SORA library is hosted on a repository. It was developed with support of the , that agglomerates the efforts of the Paris, Granada, and Rio teams [1]. The is funded by the ERC (European Research Council) under the European Community's H2020 (2014-2020/ERC Grant Agreement No. 669416). Also, this project is supported by (Laboratório Interinstitucional de e-Astronomia), INCT do e-Universo (CNPQ grants 465376/2014-2), by FAPESP (proc. 2018/11239-8), by CNPQ (proc. 300472/2020-0, 150612/2020-6), and by CAPES-PRINT/UNESP (88887.571156/2020-00) in Brazil.

Figure 2: The SORA team thanks the following institutions, groups and projects for their support: ERC Lucky Star, LIneA/Brazil, INCT do e-Universo/Brazil, UNESP/Brazil, GDOP/Brazil, Observatoire de Paris/France, Observatório Nacional/Brazil and CNPq/Brazil.

[1] The Paris, Granada, and Rio teams are professionals astronomers affiliated mainly in the following institutions:

- LESIA - Observatoire de Paris, France;
- Institut Polytechnique des Sciences Avancées, France;
- IMCCE - Observatoire de Paris, France;
- Instituto de Astrofísica de Andalucía, Spain;
- Laboratório Interinstitucional de e-Astronomia, Brazil;
- INCT do e-Universo, Brazil;
- Observatório Nacional/MCTI, Brazil;
- Federal University of Technology - Paraná, Brazil;
- UNESP - São Paulo State University, Brazil;
- Universidade Federal do Rio de Janeiro - Observatório do Valongo, Brazil;

CHAPTER

TWO

USING AND CITING SORA

2.1 Citing

SORA is a free software, and we kindly ask users that make use of it (or of part of it), especially in projects that results in scientific publications, to include the following citation:

```
Gomes-Júnior et al. (2022). SORA: Stellar occultation reduction and analysis.
MNRAS, Volume 511, Issue 1, March 2022, Pages 1167-1181
DOI: 10.1093/mnras/stac032
```

The scientific documentation of SORA is described in the paper available on Monthly Notices of the Royal Astronomical Society, which can be accessed on the following links:

- ;
- ;
- ;

Also, for a scientific publication, please be attentive the following terms of use for co-authorship:

1. If you *participate in the collaboration Rio/Granada/Paris* and use SORA for the analysis of *objects within the collaboration* (e.g. TNOs, Centaurs, giant planets Trojans, among others), the SORA team may have the possibility to decide if any of its PIs should be co-author of the publication. In this case, the SORA team will commit to:
    - Teach the functionalities of the SORA package in a hands-on session or individually;
    - Run SORA, if necessary;
    - Fix any bug - in SORA code - with urgency;
    - Implement new requested features with high priority.
2. If you *participate in the collaboration Rio/Granada/Paris* and use SORA for the analysis of *objects outside the collaboration* (e.g. Main-Belt Asteroids, NEAs, among others), the SORA team will not demand the possibility to become a co-author, but the code should be properly cited as stated above.
3. If you are *NOT from the Rio/Granada/Paris collaboration* and use SORA for the analysis of *any object*, the SORA team will not demand the possibility to become a co-author, but the code should be properly cited as stated above.

As a drawback for itens 2 and 3, the SORA team will NOT commit to:

- Teach the functionalities of the SORA package in a hands-on session or individually;
- Run SORA;
- Fix any bug - in SORA code - with urgency, unless it is a critical bug;

- Implement new requested features with high priority;

In the event of granting priority or special attention to the user by the SORA team, further discussions will be made to address the addition of a specific co-author (or co-authors) from the team in the publication.

2.2 License

CHAPTER

THREE

INSTALLATION

3.1 Python package requirements

Several SORA functionalities use other Python-based libraries. Below are listed the library dependencies and their minimal version needed to use SORA. Most of those packages are installed on the fly using the *pip install* method, except for Cartopy.

- Astropy (4.3.1): For astronomical related functions, mainly coordinates and time.
- Astroquery (0.4.6): To query astronomical database as JPL and Vizier.
- Matplotlib (3.5.3): For easy and beautiful plots.
- NumPy (1.21): Otimized mathematical functions.
- SciPy (1.7.1): Otimized functions for mathematics, science, and engineering.
- SpiceyPy (5.1.1): SPICE/NAIF functions in python.
- PyERFA (2.0): Python wrapper for the ERFA library based on the SOFA library.
- Cartopy (0.20.4): Geospatial data processing to produce maps.
- Shapely (1.8.2): Package for set-theoretic analysis and manipulation of planar features.

3.2 Installing SORA

If you are new to Python or not familiar with Python virtual environments, we recommend starting by installing the Anaconda Distribution. This works on all platforms (Linux, macOS, Windows) and installs a full-featured scientific Python in a user directory without requiring root permissions. For a better experience with SORA, we recommend the use of Jupyter. The creation of a dedicated Conda environment for SORA is suggested to avoid requirement issues.

The user can install SORA and most of its requirements using **pip**, only Cartopy should be installed by hand afterwards.

```
>>> user@path$> pip install sora-astro
>>> user@path$> conda install -c conda-forge cartopy
```

If you are a user, you can also use:

```
>>> user@path$> git clone https://github.com/riogroup/SORA/sora.git
>>> user@path$> cd sora
>>> user@path$> pip install .
>>> user@path$> conda install -c conda-forge cartopy
```

When new versions are available, the user can update it downloading the last release from the SORA package in the riogroup organisation on . If you want to be notified just follow the package.

3.3 Functionalities

With SORA, among other more advanced tasks, the user can easely:

1. Predict stellar occultations and obtain predictions maps;
2. Check when a stellar occultation will happen for a given observer;
3. Analyse occultation light curves and determine the immersion and emersion times for the event;
4. Plot and check the chords in the skyplane;
5. Fit a circle for events with less than 3 chords or an ellipse for events with more chords;
6. Determine the astrometric position of the occulting object, its apparent size and projected shape.

All these steps can be found in our Jupyter-Notebooks Tutorials.

CHAPTER

FOUR

RELEASES

4.1 SORA v0.3 (2023/Jan/31)

4.1.1 New Features

sora.body

- Added *get_position()* in Body as a shortcut to *ephem.get_position()*. [#76]
- Created new *frame* submodule: [#74]
  - Define a new class for the orientation of the body (*PlanetocentricFrame*), giving its rotation velocity, pole coordinates, and precession parameters as defined by Archinal et al. (2018). With it, we can transform to other reference frames using Astropy.
  - The definitions of the orientation parameters in Archinal et al. (2018) are hardcoded. The user can overwrite it by providing its own frame.
  - With the *PlanetocentricFrame*, we process and rotate the body to find its orientation at any epoch (must consider light time).
- Created new *shape* submodule to: [#74]
  - Define the new *Shape3D* class where we can provide to the body a wavefront file with its 3D shape. An image corresponding to the texture can also be provided.
  - Define the new *Ellipsoid* class where the 3-dimensional sizes of the ellipsoid are provided to create a 3D object.
  - Plot the 3D shape on the tangent plane considering the sub-observer coordinates, its pole orientation, and the solar illumination.
  - Compute the limb of the rotated 3D shape to use it in the fitting process. This limb now requires a dependency on shapely.

sora.lightcurve

- Added argument "method" in *LightCurve.occ_lcfit()* method to allow the use of different fitting procedures. [#75]
- Added argument "threads" in *LightCurve.occ_lcfit()* method to allow shared memory parallel processing. [#75]
- Added argument "verbose" in *LightCurve.occ_lcfit()* method to enable progress bar during fitting procedure. [#75]

sora.occultation

- Added argument "method" in *Occultation.fit_ellipse()* method to allow the use of different fitting procedures. [#75]
- Added argument "threads" in *Occultation.fit_ellipse()* method to allow shared memory parallel processing. [#75]
- Added argument "verbose" in *Occultation.fit_ellipse()* method to enable progress bar during fitting procedure. [#75]
- New method *fit_shape()* is created to fit the occultation chords to the 3D shape. [#74]
- An auxiliary fitting function *fit_to_limb()* was created to fit the chords to a limb provided by the user. [#74]

sora.prediction

- Included the `VizierCatalogue` in the prediction function. [#71]
- Defined Gaia-DR3 as the default in the *prediction()* function. [#76]

sora.star

- Added a new class called `VizierCatalogue` to allow the definition and usage of any catalogue available on Vizier. [#71]
- Included the `VizierCatalogue` in the Star class. [#71]
- Included the Gaia-DR3 catalogue, released on June 13, 2022, in the catalogue list. [#76]
- Defined Gaia-DR3 as the default in the Star class. [#76]

sora.stats

- New module dedicated to design model fitting procedures using different techniques. [#75]

4.1.2 API Changes

sora.ephem

- *id_type* for planets and satellites are passed as *None* instead of *major_body*, following changes made in Horizons. [#76]

4.1.3 Bug Fixes

sora.ephem

- Fixed bug that did not calculated ephemeris for observer when using Horizons if the observer did not have IAU code. [#80]

4.2 SORA v0.2.1 (2021/Aug/30)

4.2.1 New Features

sora.body

- Added function "Body.to_log()". [#65]

sora.extra

- Added function "ChiSquare.to_log()". [#65]

sora.observer

- Added function "Observer.to_log()". [#65]

sora.occultation

- Added plot function "Occultation.plot_radial_dispersion()". [#65]
- Added functions that calculates the geometric albedo [#65]

sora.prediction

- Added new parameter "site_box_alpha" to plot map. [#65]

sora.star

- Added function "Star.to_log()". [#65]
- Added function that corrects the Gaia-EDR3 proper motion as suggested by Cantat-Gaudin and Brandt (2021) [#65]

4.2.2 API Changes

sora.occultation

- Occultation.check_velocities() now calculates the normal velocity considering the ellipse fitted in fit_ellipse replacing the radial velocity. [#66]

sora.star

- Added warning for Gaia stars with a high RUWE or Duplicity flag, indicating a possible issue with this star's astrometry. [#65]

4.2.3 Bug Fixes

- Updated versions of dependencies to avoid bug caused by conflict between them [#67]

sora.body

- Fixed wrong albedos in satdb. [#66]

sora.ephem

- Added argument "meta" in "EphemHorizons", thus fixing a bug when predicting with this class. [#65]
- Fixed bug when radius is inputted in the ephem classes. [#67]

sora.lightcurve

- Debuging "LightCurve.to_file()" and "LightCurve.plot_model()". [#65]
- Fixed bug where LightCurve did not read occultation parameters if a flux or file was not provided. [#66]

sora.occultation

- Fixed bug that overwrote occultation parameters in the LightCurve when added to the ChordList. [#66]
- Fixed bug that prevented the user to ignore the chord name as labels when plotting the chords. [#66]
- Fixed bug that prevented Occultation instantiation if the size of the star can not be determined [#67]

sora.prediction

- Added white "facecolor" to occultation maps. [#65]
- Fixed MAJOR bug that inverted the shadow velocity in some cases. [#66]
- Fixed bug in the occultation map that did not plot the direction arrow in some cases when cartopy>=0.18. [#67]

4.3 SORA v0.2 (2021/Jun/14)

4.3.1 New Features

sora.body

- Created new Body Class which downloads the occulting body information from online source. At the moment, it downloads only from the Small-Body DataBase. The Body class will be the manager for all the Body information, such as Ephem, Shape, Ring, etc. [#51]
- New Class PhysicalData, which inherits from astropy.units.quantity.Quantity, is created to handle physical data with uncertainty, reference and notes. [#51]
- "pole_position_angle" and "apparent_magnitude" functions are now present in Body instead of Ephem.[#51]
- Created a hardcoded satellite database to complement missing data of SBDB. It must be replaced in the future. [#61]

sora.ephem

- A new EphemHorizons was created which is strictly equal to EphemJPL (EphemJPL may be removed in v1.0). [#51]
- A new function that downloads the kernel from JPL was added. [#33]

sora.extra

- Allow two ChiSquare objects to be combined into one: *chi3 = chi1 + chi2* [#61]
- New function get_ellipse_points() that calculates the positions on the perimeter of an ellipse [#60]

sora.observer

- New Spacecraft class developed to handle the geometry of a spacecraft observation. To use it,it is necessary a spkid and ephemeris. Ex: *spacecraft = Spacecraft(name='New Horizons', spkid='-98', ephem='horizons')*. [#63]
- The Observer class was updated to have an ephemeris as well. [#63]
- Now the observer can be passed as parameter to *Ephem*.get_position(observer=observer)*, *Star.get_position()*, *Body.get_pole_position_angle()* and *Body.apparent_magnitude()*. [#63]

sora.occultation

- A shortcut was created in Occultation where the user can pass the coordinate of the star directly to Occultation, the Star object will be created automatically. [#46]
- New Chord Class introduced to handle a chord with an Observer and a LightCurve. [#53]
- New ChordList Class introduced to handle the list of Chords in an Occultation. [#53]
- New function .get_impact_param() that calculates the impact parameter, minimal distance between the chord and the centre position, in Chord and ChordList.[#60]
- New function .get_theoretical_times(), that calculates the theoretical times and chord size for a given ellipse in Chord and ChordList. [#60]

- New function .check_time_shift() that calculates the offset in time to align the center of the chords in Occultation. [#60]
- New parameters sigma_result, that saves the result with an extended error bar, and ellipse_error, that adds a further systematic error to be considered, in Occultation.fit_ellipse(). [#60]
- New function filter_negative_chord() that compares the ChiSquare from an Ellipse fitting with the chords and remove the solutions that would cross a negative chord [#60]
- New method to calculate the "f" and "g" positions for observers without referring to the geocenter. [#63]

sora.prediction

- prediction() now makes use of the user input of the star to calculate faster the occultation parameters. [#48]
- prediction() now can make predictions using Gaia-EDR3. A new parameter "catalogue" was created for choosing between Gaia-DR2 and Gaia-EDR3.[#61]
- Fixed bug when plotting the heights in the map in a rotated projection. [#54]
- prediction() can now predict for any observer. Ex: *prediction(..., reference_center=observer)*. [#63]

sora.star

- A new method get_position() was implemented in Star() that will replace geocentric() and barycentric() methods [#63]

4.3.2 API Changes

- Update the argument "log" to "verbose" on all modules. [#61]

sora.ephem

- "pole_position_angle" and "apparent_magnitude" is passed to Body Class. In Ephem, it will raise a FutureWarning. [#51]
- The Ephem classes are now passed through the Body Class which will have priority over Ephem attributes. Parameters such as "spkid", "radius", "H" and "G". [#51]
- All Ephem Classes now inherits from BaseEphem, which holds core functionality for all of them. [#51]

sora.lightcurve

- Removed the necessity for LightCurve to have a unique name associated. [#53]
- Cycle time is now determined via mode instead of median. [#56]

sora.observer

- Removed the necessity for Observer to have a unique name associated. [#53]

sora.occultation

- The new Body Class was implemented in Occultation. For backward compatibility, the previous usage is still possible if the Ephem object have a name. The Body Class is only required if the object is a planet or a planetary satellite. [#51]
- Deprecated some functions that were passed to ChordList. [#53]

sora.prediction

- prediction() now creates the time array inside each division to avoid memory overflow. [#48]
- prediction() now propagates the positions of the stars using only the proper motions before comparing the stars with the ephemeris. [#48]
- The new Body Class was implemented in prediction. For backward compatibility, the previous usage is still possible. [#51]

4.3.3 Bug Fixes

sora.lightcurve

- Corrected bug in LightCurve model where the size of the star was being interpreted as radius instead of diameter [#60]

sora.prediction

- Fixes issue that happened in occ_params() when the instant of the occultation was outside the given range. The function now gives appropriate error messages. The automatic range search was increased to 50 min from central instant in a recursive search. [#45#48]

4.4 SORA v0.1.2 (2020/Dec/14)

4.4.1 New Features

sora.star

- Star() is now able to fully receive astrometric parameters from the user. [#48]
- Star() is able to download and use the distance from Bailer-Jones et al (2018). [#27]
- Gaia-EDR3 was implemented in Star() and is now a default feature. [#52]

4.4.2 API Changes

sora.star

- The star module was moved to its own directory. [#52]

4.4.3 Bug Fixes

sora.star

- Star now calculates the robust propagation of the position of the star and correspondent uncertainties. [#18]
- Fixed bug in Star().__str__() where pmDEC was printed wrong. [#43]
- A small bug fix was made in Star with the units of the star position error when coordinates are local. [#51]

4.5 SORA v0.1.1 (2020/Jul/30)

4.5.1 New Features

sora.config

- Module to verify if kwargs are allowed was created. This was included throughout the code. [#8]

sora.extra

- Added a parameter that allows the used to plot a dot corresponding the center of the ellipse [#35]

sora.lightcurve

- Property LightCurve.time_mean that returns the mean time of the chord (positive) or the mean time of the observation (negative). [#34]

sora.observer

- Function Observer.altaz() that calculates the altitude and azimuth for a given target and instant. [#34]

sora.prediction

- Four new parameters were added to *plot_occ_map()*: *path*: for the user to select a directory where to save the plots; *site_name*: If True, the name of the sites will be plotted; *chord_delta* and *chord_geo*: for the user to plot the path of a chord from distance of the center or passing by some coordinate, respectively. [#35]
- Two methods were added to *PredictionTable()* to help the user to remove bad events from table: *keep_from_selected_images()* and *remove_occ()*. [#35]

4.5.2 API Changes

sora.config

- config module is now a directory. It now includes a module with decorators and another for variables. [#31#35]

sora.ephem

- In EphemKernel, *code* argument was replaced by *spkid*. When using 'code', a FutureWarning is raised stating *code* as deprecated and will be removed from v1.0. [#26]

sora.lightcurve

- In LightCurve.immersion and LightCurve.emersion, an error will rise when these values were not instanciated or fitted. [#34]
- Now the user has the possibility to redefine *tref*, *immersion*, *emersion*, *initial_time* and *end_time* after instantiated. [#35]
- *lambda_0* argument was replaced by *central_bandpass* and *delta_lambda* by *delta_bandpass*. When using 'lambda_0' or *delta_lambda*, a FutureWarning is raised stating *lambda_0* or *delta_lambda* as deprecated and will be removed from v1.0. [#36]

sora.occultation

- Occultation.new_astrometric_positions() now shows a warning when time is far by more than 1 day from the occultation closest approach. [#21]
- Occultation.to_log() and print(Occultation) added the polar radius, equivalent radius, the Sun-Geocenter-Target angle and the Moon-Geocenter-Target angle, geocentric albedo, the altitude and azimuth of the target for each Observer. [#17]
- In *fit_ellipse()*, *pos_angle* and *dpos_angle* were deprecated in favor of *position_angle* and *dposition_angle*. [#35]
- Changed "GCRS" to "Geocentric" in the string representation to avoid confusion about the reference frame. [#35]

sora.prediction

- prediction() now calculates the ephemeris inside each division to avoid memory overflow. [#31]
- PredictionTable.to_ow() will now raise a warning if the radius or the error of the ephemeris is not present. [#35]

sora.star

- Now Star downloads all parameters from Gaia and saves them in the *meta_gaia* attribute [#35]

4.5.3 Bug Fixes

sora.ephem

- Added function get_position() to EphemPlanete. This corrects a bug that prevented Occultation to run with EphemPlanete. [#41]
- Fixed bug in EphemJPL where *id_type* was redefined inside __init__(). [#41]

sora.lightcurve

- Fixed error that appears when the fit was done separately (immersion and emersion times). Now the final model agrees with the fitted values. [#9]
- Fixed error when the file with the light curve has three columns. [#19]
- Fixed error when the exptime within the LightCurve was set as zero or negative. [#23]
- Fixed error in the automatic mode of LightCurve.normalize(). [#34]
- Fixed bug that was raised in LightCurve.log() when there were no initial or end times for lightcurves instantiated with immersion and emersion. [#35]

sora.occultation

- Corrected error calculation using err = sqrt(star_err^2 + fit_err^2) [#18]
- Occultation.plot_occ_map() now uses the fitted ellipse to calculate the projected shadow radius [#22]
- Corrected bug that raised an error when calling Occultation.get_map_sites() and there were no observation added to Occultation. [#31]
- Corrected bug that did not save the fitted params in all occultations when more than one occultation was used in fit_ellipse(). [#35]
- Added *axis_labels* and *lw* (linewidth) to Occultation.plot_chords(). [#35]

sora.prediction

- Fixed error that was generated when only one prediction was found. [#16]
- Fixed error in the output format of PredictionTable.to_ow() when coordinate was positive [#35]

4.6 SORA v0.1 (2020/May/20)

4.6.1 Classes

The documentation of all classes and functions are on their docstrings, while the scientific part is presented in the full documentation. Here follows a list with the main Classes:

Ephem Three Classes created to generate geocentric ephemeris for a given solar system object. **EphemJPL** queries the JPL Horizons service and download ephemeris information. **EphemKernel** reads the BSP files to calculate the ephemeris using the Spiceypy package. **EphemPlanet** reads an ASCII file with previously determined positions and interpolate them for a given instant.

JPL Horizons - https://ssd.jpl.nasa.gov/horizons.cgi

Star Class created to deal with the star parameters. From the Gaia-DR2 Source ID or a sky region, it queries the VizieR service and downloads the star's information. From Gaia DR2 Catalog (Gaia Collaboration 2016a, 2016b and 2018) it gets the RA, DEC, parallax, proper motions, G magnitude and star radius; from the NOMAD Catalog (Zacharias et al. 2004) it gets the B, V, R, J, H and K magnitudes. The user can calculate the ICRS coordinate of the star at any epoch. It can be barycentric (corrected from proper motion) or geocentric (corrected from proper motion and parallax). Also, the apparent diameter of the star is calculated using Gaia DR2 information, or some models such as Van Belle (1999) and Kervella et al. (2004).

Gaia - Gaia Collaboration 2016a, 2016b and 2018 Mission: https://ui.adsabs.harvard.edu/abs/2016A%26A...595A...1G/abstract DR1: https://ui.adsabs.harvard.edu/abs/2016A%26A...595A...2G/abstract DR2: https://ui.adsabs.harvard.edu/abs/2018A%26A...616A...1G/abstract VizieR - https://vizier.u-strasbg.fr/viz-bin/VizieR NOMAD - Zacharias et al. 2004 https://ui.adsabs.harvard.edu/abs/2004AAS...205.4815Z/abstract Van Belle, 1999 - https://ui.adsabs.harvard.edu/abs/1999PASP..111.1515V/abstract Kervella, 2004 - https://ui.adsabs.harvard.edu/abs/2004A%26A...426..297K/abstract

Observer: Object Class created to deal with the observer location. The user can also download the ground-based observatories from the Minor Planet Center (MPC) database.

MPC sites - https://minorplanetcenter.net/iau/lists/ObsCodesF.html

Light Curve: Object Class that receives the observational light curve (with time and the occulted star normalized photometry relative to reference stars) and some observational parameters (filter and exposure time). It has functions to determine the instants that the solar system object enters in front of the star and leaves, (immersion and emersion times, respectively). The model considers a sharp-edge occultation model (geometric) convolved with Fresnel diffraction, stellar diameter (projected at the body distance) and finite integration time (Widemann et al., 2009; Sicardy et al., 2011).

Widemann et al. 2009 - https://ui.adsabs.harvard.edu/abs/2009Icar..199..458W/abstract Sicardy et al. 2011 - https://ui.adsabs.harvard.edu/abs/2011Natur.478..493S/abstract

Occultation: Main Object Class within SORA, created to analyze stellar occultations, and control all the other Object Classes within this package. Its functions allow converting the times for each observatory in the occulted body positions in the sky plane relative to the occulted star (f, g) (IERS Conventions). Also, to obtain the best ellipse parameters (centre position, apparent equatorial radius, oblateness and the position angle of the apparent polar radius) that fit the points. The results are the apparent size, shape and astrometrical position of the occulting body.

IERS Conventions: https://www.iers.org/IERS/EN/Publications/TechnicalNotes/tn36.html

Some extra Objects Classes:

PredictionTable: Using the **prediction** function within SORA results in an Object Class that is a slight modification of an AstropyTable. The added changes allow to create the occultation map for each prediction, convert into specific formats, such as OccultWatcher and PRAIA (Assafin et al. (2011)).

OccultWatcher - https://www.occultwatcher.net/ Assafin et al., 2011 - https://ui.adsabs.harvard.edu/abs/2011gfun.conf...85A/abstract

ChiSquare: This Object Class is the result of the fitting functions within SORA, such as _LightCurve.occ_lcfit()_ and _Occultation.fit_ellipse()_. This Class has functions that allow viewing the values that minimize the χ^2 tests, the uncertainties within $n\sigma$, plotting the tests, and saving the values.

4.6.2 INPUTS AND OUTPUTS

INPUTS

- **Event Related (Star and Ephem)**
  - Object Name or provisory designation
  - Object Code (only for EphemKernel)
  - BSP file and name (only for EphemKernel)
  - DE file and name (only for EphemKernel)
  - Ephemeris offset for RA and DEC - $\Delta\alpha \cdot \cos\delta$, $\Delta\delta$ (set as 0,0)
  - Occultation date and time
  - Occulted star coordinates RA and DEC; or Gaia code
  - Star offset for RA and DEC - $\Delta\alpha \cdot \cos\delta$, $\Delta\delta$ (set as 0,0)
- **Observer Related**
  - Site name and location (latitude, longitude, and height; or IAU/MPC code)
  - Light curve file and name; or array with fluxes and times; or immersion and emersion times
  - Exposure time in seconds
  - Observational bandwidth in microns (set as 0.7 ± 0.3 microns, Clear)
- **Fitting Related**
  - Initial guess for light curve fitting: immersion, emersion and opacity.
  - Range to explore all three parameters
  - Initial guess for ellipse parameters: center (f,g), equatorial radius, oblateness, and position angle
  - Range to explore all five parameters

OUTPUTS

- Star
  - Star Gaia-DR2 ID
  - Star coordinates at 2015.5 and uncertainty - RA and DEC (hh mm ss.sss , +dd mm ss.sss, mas, mas)
  - Star proper motion - in RA, DEC - and uncertainties (mas/yr)
  - Star parallax and uncertainty (mas)
  - Star coordinates propagated to event epoch and uncertainty - RA and DEC (hh mm ss.sss , +dd mm ss.sss, mas, mas)
  - Star magnitudes G, B, V, R, J, H, K (mag)
  - Star projected diameter and model (km and mas, model: GDR2, Van Belle, Kervella)
  - Star offset applied in RA and DEC (mas, mas)
- Object and Ephemeris
  - Object Name

  - Object radius (km)
  - Object mass (kg)
  - Ephemeris kernel (version and DE)
  - Offset applied in RA/DEC (mas, mas)
  - Object's distance (AU)
  - Object apparent magnitude for the date (mag)
- Occultation
  - Event date and time (yyyy-mm-dd hh:mm:ss.sss)
  - Closest approach Angle - CA (arcsec)
  - Reference time (yyyy-mm-dd hh:mm:ss.sss)
  - Position Angle - PA (degree)
  - Shadow's velocity relative to the geocenter (km/s)
  - Number of positive observations
  - Number of negative observations
- Observer Information
  - Detection status (positive, negative, overcast, tech. problem, other)
  - Site Name
  - Site MPC/IAU code (if any)
  - Site coordinates - Latitude, Longitude and height (dd mm ss.s ; dd mm ss.s ; m)
  - Light curve file name
  - Number of images (lines in LC)
- Light curve fitting information (for each positive detection)
  - Acquisition start time (yyyy-mm-dd hh:mm:ss.sss)
  - Acquisition end time (yyyy-mm-dd hh:mm:ss.sss)
  - Exposure time (s)
  - Cycle time (s)
  - Time offset applied in LC (s)
  - Light curve calculated RMS
  - Calculated normalised flux and bottom flux (standard = 1, 0)
  - Band width and uncertainty (microns)
  - Shadow's velocity relative to the station (km/s)
  - Fresnel scale (s and km)
  - Projected stellar size scale (s and km)
  - Integration time scale (s and km)
  - Dead time scale (s and km)
  - Model resolution - size of synthetic LC point (s and km)

  - Immersion Time and uncertainty (yyyy-mm-dd hh:mm:ss.sss +/- s.sss)
  - Immersion Time and uncertainty - 1σ and 3σ (s)
  - Emersion Time and uncertainty (yyyy-mm-dd hh:mm:ss.sss +/- s.sss)
  - χ^2 fit model
  - Emersion Time and uncertainty - 1σ and 3σ (s)
  - Minimum Chi-square - χ^2_{min}
  - Number of fitted points for im- and emersion
  - Number of fitted parameters
  - Minimum Chi-square per degree of freedom - $\chi^2_{min-pdf}$
- Elipse fit procedure
  - Fitted parameters: Equatorial radius and uncertainty (km); Center position (f_0, g_0) and 1σ uncertainties (km, km); Oblateness and uncertainty; Position angle and uncertainty (degree)
  - Minimum Chi-square - χ^2_{min}
  - Minimum Chi-square per degree of freedom - $\chi^2_{min-pdf}$
  - Number points used to fit (X points from Y chords)
  - Astrometric object center position at occ. time and uncertainty (hh mm ss.sss +dd mm ss.sss $\pm$ mas)
- Plots and files (some are optional)
  - Prediction map (Lucky Star model)
  - Normalised light curve - for each site (x = time; y = flux)
  - Chi-square map for immersion and emersion times (x = time; y = χ^2)
  - Light curve and synthetic LC- for each site (x = time; y = flux)
  - Chords projected in sky plane (x = ξ (km); y = η (km))
  - Chi-square map for each ellipse parameter (x = time; y = χ^2_{param})
  - Chords projected in sky plane and the best ellipse fitted with 1σ uncertainties (x = ξ (km); y = η (km))
  - Log file with all information

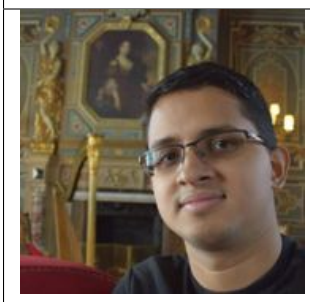
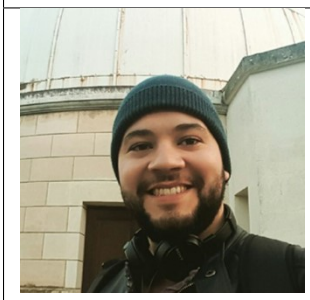
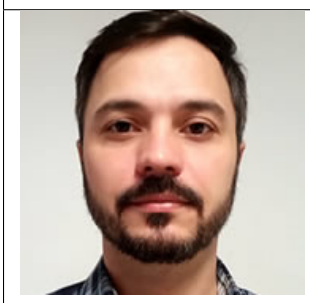
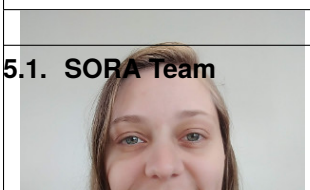
CHAPTER

FIVE

ABOUT US

The SORA package was developed with support of the , that agglomerates the efforts of the Paris, Granada, and Rio teams. The is funded by the ERC (European Research Council) under the European Community's H2020 (2014-2020/ERC Grant Agreement No. 669416). Also, this project is supported by (Laboratório Interinstitucional de e-Astronomia), INCT do e-Universo (CNPQ grants 465376/2014-2), by FAPESP (proc. 2018/11239-8) and by CNPQ (proc. 300472/2020-0), Brazil.

5.1 SORA Team

Photo	Details
	**Dr. Altair Ramos Gomes Júnior** Assistant Professor Federal University of Uberlândia/Brazil
	**Dr. Bruno Eduardo Morgado** Assistant Professor UFRJ-Observatório do Valongo and LIneA/Brazil
	**Dr. Gustavo Benedetti Rossi** Postdoctoral Researcher UNESP-Campus Guaratinguetá and LIneA/Brazil
	**Dr. Rodrigo Carlos Boufleur** Postdoctoral Researcher Observatório Nacional and LIneA/Brazil
	

5.2 Contact us

Any question or suggestion can be added to our Issue webpage or addressed to our mailbox **rio.occteam@gmail.com**. The SORA team can also be contacted through Slack. In this case, you can join our Slack workspace . The SORA team thanks for your suggestions and we are looking forward to improve our code with yours insights.

CHAPTER

SIX

MODULES

SORA is a Python-based, object-oriented library for optimal analysis of stellar occultation data. The user can use this library to build pipelines to analyse their stellar occultation's data. Here follows the details for each module: **body**, **ephem**, **star**, **prediction**, **observer**, **lightCurve**, **occultation** and **extra**.

6.1 sora.body

6.1.1 The Body Class

class sora.**Body**(*name*, *database='auto'*, ***kwargs*)

Class that contains and manages the information of the body.

name

The name of the object. It can be the used *spkid* or *designation number* to query the SBDB (Small-Body DataBase). In this case, the name is case insensitive.

Type
str, required

database

The database to query the object. It can be `satdb` for our temporary hardcoded satellite database, or `'sbdb'` to query on the SBDB. If database is set as `auto` it will try first with `satdb`, then `sbdb`. If the user wants to use their own information, database must be given as `None`. In this case, *spkid* parameter must be given.

Type
str, optional, default='auto'

ephem

An Ephem Class that contains information about the ephemeris. It can be "horizons" to automatically defined an EphemHorizons object or a list of kernels to automatically define an EphemKernel object.

Type
sora.EphemKernel, *sora.EphemHorizons*, *sora.EphemJPL*, *sora.EphemPlanete*

orbit_class

It defines the Orbital class of the body. It can be `TNO`, `Satellite`, `Centaur`, `comet`, `asteroid`, `trojan`, `neo`, and `planet`. It is important for a better characterization of the object. If a different value is given, it will be defined as `unclassified`.

Type
str

`spkid`

If `database=None`, the user must give a *spkid* or an *ephem* which has the *spkid* parameter.

Type
str, *int*, *float*

`shape`

It defines the input shape of the body. It can be a body.shape object or the path to OBJ file.

Type
str, *sora.body.shape.Shape3D*

`albedo`

The albedo of the object.

Type
float, *int*

`H`

The absolute magnitude.

Type
float, *int*

`G`

The phase slope.

Type
float, *int*

`diameter`

The diameter of the object, in km.

Type
float, *int*, *astropy.quantity.Quantity*

`density`

The density of the object, in g/cm^3.

Type
float, *int*, *astropy.quantity.Quantity*

`GM`

The Standard Gravitational Parameter, in km^3/s^2.

Type
float, *int*, *astropy.quantity.Quantity*

`rotation`

The Rotation of the object, in hours.

Type
float, *int*, *astropy.quantity.Quantity*

`pole`

The Pole coordinates of the object. It can be a *SkyCoord object* or a string in the format `'hh mm ss.ss +dd mm ss.ss'`.

Type
str, *astropy.coordinates.SkyCoord*

BV

The B-V color.

Type
float, *int*

UB

The U-B color.

Type
float, *int*

smass

The spectral type in SMASS classification.

Type
str

tholen

The spectral type in Tholen classification.

Type
str

Note: The following attributes are are returned from the Small-Body DataBase when `database='sbdb'` or from our temporary hardcoded Satellite DataBase when `database='satdb'`:

orbit_class, *spkid*, *albedo*, *H*, *G*, *diameter*, *density*, *GM*, *rotation*, *pole*, *BV*, *UB*, *smass*, and *tholen*.

These are physical parameters the user can give to the object. If a query is made and user gives a parameter, the parameter given by the user is defined in the *Body* object.

__from_local(*name*, *spkid*)

Defines Body object with default values for mode='local'.

__from_sbdb(*name*)

Searches the object in the SBDB and defines its physical parameters.

Parameters
name (*str*) – The *name*, *spkid* or *designation number* of the Small Body.

apparent_magnitude(*time*, *observer='geocenter'*)

Calculates the object's apparent magnitude.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the object's apparent magnitude. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **observer** (*str*, *sora.Observer*, *sora.Spacecraft*) – IAU code of the observer (must be present in given list of kernels), a SORA observer object or a string: ['geocenter', 'barycenter']

Returns
ap_mag – Object apparent magnitude.

Return type
float

`get_pole_position_angle`(*time*, *observer='geocenter'*)

Returns the pole position angle and the aperture angle relative to the geocenter.

Parameters

- **time** (*str*, *astropy.time.Time*) – Time from which to calculate the position. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **observer** (*str*, *sora.Observer*, *sora.Spacecraft*) – IAU code of the observer (must be present in given list of kernels), a SORA observer object or a string: ['geocenter', 'barycenter']

Returns

position_angle, aperture_angle – Position angle and aperture angle of the object's pole, in degrees.

Return type

float array

`get_position`(*time*, *observer='geocenter'*)

Returns the object geocentric position.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the object position. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **observer** (*str*, *sora.Observer*, *sora.Spacecraft*) – IAU code of the observer (must be present in given list of kernels), a SORA observer object or a string: ['geocenter', 'barycenter']

Returns

coord – Astropy SkyCoord object with the object coordinates at the given time.

Return type

astropy.coordinates.SkyCoord

`plot`(*time=None*, *observer='geocenter'*, *center_f=0*, *center_g=0*, *contour=False*, *ax=None*, *plot_pole=True*, ***kwargs*)

Plots the body shape as viewed by observer at some time given the body orientation. If the user wants to dictate the orientation, please use *shape.plot()* instead.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the object's apparent magnitude. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object. It must be only one value.
- **observer** (*str*, *sora.Observer*, *sora.Spacecraft*) – IAU code of the observer (must be present in given list of kernels), a SORA observer object or a string: ['geocenter', 'barycenter']
- **center_f** (*int*, *float*) – Offset of the center of the body in the East direction, in km
- **center_g** (*int*, *float*) – Offset of the center of the body in the North direction, in km
- **radial_offset** (*int*, *float*) – Offset of the center of the body in the direction of observation, in km
- **ax** (*matplotlib.pyplot.Axes*) – The axes where to make the plot. If None, it will use the default axes.

- **contour** (*bool*) – If True, it plots the limb of the projected shape. If False, it plots the 3D shape. Default: False.
- **plot_pole** (*bool*) – If True, the direction of the pole is plotted. Ignored if *contour=True*

to_log(*namefile*)

Saves the body log to a file.

Parameters

namefile (*str*) – Filename to save the log.

6.1.2 PhysicalData Class

class sora.body.**PhysicalData**(*name*, *value*, *uncertainty=0.0*, *reference='User'*, *notes=''*, *unit=Unit(dimensionless)*, *raise_error=False*)

Defines PhysicalData with uncertainty, reference and notes.

Note: It inherits from astropy.units.quantity.Quantity().

Parameters

- **name** (*str*) – The name representing the corresponding physical parameter.
- **value** (*int*, *float*, *str*, *~numpy.ndarray*, *astropy.quantity.Quantity*) – The numerical value of this quantity in the units given by unit. If a *Quantity* (or any other valid object with a `unit` attribute), creates a new *Quantity* object, converting to *unit* units as needed. If a string, it is converted to a number or *Quantity*, depending on whether a unit is present.
- **uncertainty** (*int*, *float*, *str*, *~numpy.ndarray*, *astropy.quantity.Quantity*, default=0) – The numerical value of this quantity in the units given by unit. If a *Quantity* (or any other valid object with a `unit` attribute), creates a new *Quantity* object, converting to *unit* units as needed. If a string, it is converted to a number or *Quantity*, depending on whether a unit is present.
- **reference** (*str*, default='User') – A string stating the reference for the parameter value.
- **notes** (*str*, default='') – Any other important information about the physical parameter.
- **unit** (*str*, *~astropy.units.UnitBase* instance, default='dimensionless') – An object that represents the unit associated with the input value. Must be an *~astropy.units.UnitBase* object or a string parsable by the `units` package.
- **raise_error** (*bool*, default=False) – If `value=None` or `raise_error=True` the function raises an error, else *value* is redefined to `NaN`.

6.1.3 Complementary functions

sora.body.utils.**apparent_magnitude**(*H*, *G*, *dist*, *sundist*, *phase=0.0*)

Calculates the object's apparent magnitude.

Parameters

- **H** (*int*, *float*) – Absolute magnitude.
- **G** (*int*, *float*) – Slope parameter.
- **dist** (*int*, *float*) – Observer-object distance, in AU.

- **sundist** (*int*, *float*) – Sun-object distance, in AU.
- **phase** (*int*, *float*, default=0) – Phase angle (Sun-Target-Observer), in degrees.

Returns
ap_mag – Apparent magnitude.

Return type
float

sora.body.utils.**search_sbdb**(*name*)

Searches JPL Small-Body DataBase to search for object information.

As the name implies, it searches only for Small Bodies. Planets and satellites information are not retrieved by this function.

Parameters
name (*str*) – The name of the object for the search. It can be the attributed *spkid* or *designation number*. The name is case insensitive.

Returns
sbdb – An ordered dictionary with the object information.

Return type
dict

Important: The query is not an autocomplete search, so `name='Charikl'` will not find Chariklo. If more than 1 object is found, the user is asked to select the correct one (e.g: `name='neowise'`).

6.2 sora.ephem

6.2.1 The EphemPlanete Class

class sora.**EphemPlanete**(*ephem*, *name=None*, *spkid=None*, ***kwargs*)

Class used to simulate former Fortran programs *ephem_planete* and *fit_d2_ksi_eta*.

ephem

Input file with JD (UTC), geocentric RA (deg), DEC (deg), and distance (AU).

Type
file, required

name

Name of the object to search in the JPL database.

Type
str, optional, default=None

radius

Object radius, in km.

Type
int, *float*, optional, default: online database

`error_ra`

Ephemeris RA*cosDEC error, in arcsec.

Type
int, *float*, optional, default: online database

`error_dec`

Ephemeris DEC error, in arcsec.

Type
int, *float*, optional, default: online database

`mass`

Object mass, in kg.

Type
int, *float*, optional. default=0

`H`

Object absolute magnitude.

Type
int, *float*, optional, default=NaN

`G`

Object phase slope.

Type
int, *float*, optional, default=NaN

`fit_d2_ksi_eta`(*star*, *verbose=True*)

Fits the projected position (orthographic projection) of the object in the tangent sky plane relative to a star.

Parameters

- **`star`** (*str*, *astropy.coordinates.SkyCoord*) – The coordinate of the star in the same reference frame as the ephemeris.
- **`verbose`** (*bool*, optional, default=True) – Enable log printing.

`get_ksi_eta`(*time*, *star=None*)

Returns the projected position (orthographic projection) of the object in the tangent sky plane relative to a star.

Parameters

- **`time`** (*str*, *astropy.time.Time*, required) – Reference time to calculate the position. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **`star`** (*str*, *astropy.coordinates.SkyCoord*, optional, default=None) – The coordinate of the star in the same reference frame as the ephemeris.

Returns
ksi, eta – Projected position (orthographic projection) of the object in the tangent sky plane relative to a star. `ksi` is in the North-South direction (North positive). `eta` is in the East-West direction (East positive).

Return type
float array

get_position(*time*, *observer='geocenter'*)

Returns the object's geocentric position.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the object position. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **observer** (*any*) – This parameter is present in EphemPlanete for compatibility with the remaining ephem classes. The returned positions are based on the given ephemeris despite the observer.

Returns

coord – Astropy SkyCoord object with the object coordinates at the given time.

Return type

astropy.coordinates.SkyCoord

6.2.2 The EphemHorizons Class

class sora.**EphemHorizons**(*name*, *id_type='smallbody'*, *spkid=None*, ***kwargs*)

Obtains the ephemeris from Horizons/JPL service.

Note: Web tool URL: https://ssd.jpl.nasa.gov/horizons.cgi

name

Name of the object to search in the JPL database.

Type

str, required

id_type

Type of object options: `smallbody`, `majorbody` (planets but also anything that is not a small body), `designation`, `name`, `asteroid_name`, `comet_name`, `id` (Horizons id number), or `smallbody` (find the closest match under any id_type).

Type

str, default='smallbody'

radius

Object radius, in km.

Type

int, *float*, default: online database

error_ra

Ephemeris RA*cosDEC error, in arcsec.

Type

int, *float*, default: online database

error_dec

Ephemeris DEC error, in arcsec.

Type

int, *float*, default: online database

mass

Object mass, in kg.

Type
int, *float*, default=0

H

Object absolute magnitude.

Type
int, *float*, default=NaN

G

Object phase slope.

Type
int, *float*, default=NaN

get_position(*time*, *observer='geocenter'*)

Returns the ICRS position of the object for observer.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the object position. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **observer** (*str*, *sora.Observer*, *sora.Spacecraft*) – IAU code of the observer (must be present in given list of kernels), a SORA observer object or a string: ['geocenter', 'barycenter']

Returns
coord – Astropy SkyCoord object with the object coordinates at the given time.

Return type
astropy.coordinates.SkyCoord

6.2.3 The EphemKernel Class

class sora.**EphemKernel**(*kernels*, *spkid*, *name=None*, ***kwargs*)

Gets the ephemeris from BSP kernels.

Parameters

- **name** (*str*, optional, default=None) – Name of the object to search in the JPL database.
- **spkid** (*str*, required) – *spkid* of the targeting object. Former 'code' (v0.1).
- **kernels** (*list*, required) – List of paths for kernels files.
- **radius** (*int*, *float*, optional, default: online database) – Object radius, in km.
- **error_ra** (*int*, *float*, optional, default: online database) – Ephemeris RA*cosDEC error, in arcsec .
- **error_dec** (*int*, *float*, optional, default: online database) – Ephemeris DEC error, in arcsec.
- **mass** (*int*, *float*, optional, default=0) – Object Mass, in kg.
- **H** (*int*, *float*, optional, default=NaN) – Object Absolute Magnitude.
- **G** (*int*, *float*, optional, default=NaN) – Object Phase slope.

get_position(*time*, *observer='geocenter'*)

Returns the object geocentric position.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the object position. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **observer** (*str*, *sora.Observer*, *sora.Spacecraft*) – IAU code of the observer (must be present in given list of kernels), a SORA observer object or a string: ['geocenter', 'barycenter']

Returns

coord – Astropy SkyCoord object with the object coordinates at the given time.

Return type

astropy.coordinates.SkyCoord

6.2.4 Complementary functions

sora.ephem.utils.**ephem_horizons**(*time*, *target*, *observer*, *id_type='smallbody'*, *output='ephemeris'*)

Calculates the ephemeris from Horizons.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference instant to calculate ephemeris. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **target** (*str*) – IAU (kernel) code of the target.
- **observer** (*str*) – IAU (kernel) code of the observer.
- **id_type** (*str*) – Type of target object options: `smallbody`, `majorbody` (planets but also anything that is not a small body), `designation`, `name`, `asteroid_name`, `comet_name`, `id` (Horizons id number), or `smallbody` (find the closest match under any id_type).
- **output** (*str*) – The output of data. `ephemeris` will output the observed position, while `vector` will output the Cartesian state vector, without light time correction.

Returns

coord – ICRS coordinate of the target.

Return type

astropy.coordinates.SkyCoord

Notes

If the interval of time is larger than 30 days or so, a timeout error may be raised. The maximum interval will depend on the user connection.

sora.ephem.utils.**ephem_kernel**(*time*, *target*, *observer*, *kernels*, *output='ephemeris'*)

Calculates the ephemeris from kernel files.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference instant to calculate ephemeris. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **target** (*str*) – IAU (kernel) code of the target.
- **observer** (*str*) – IAU (kernel) code of the observer.

- **kernels** (*list*, *str*) – List of paths for all the kernels.
- **output** (*str*) – The output of data. `ephemeris` will output the observed position, while `vector` will output the Cartesian state vector, without light time correction.

Returns
coord – ICRS coordinate of the target.

Return type
astropy.coordinates.SkyCoord

`sora.ephem.utils.`**getBSPfromJPL**(*identifier*, *initial_date*, *final_date*, *email*, *directory='./'*)

Downloads BSP files from JPL database.

BSP files, which have information to generate the ephemeris of the objects, will be downloaded and named as (without spaces): '[identifier].bsp'.

Important: It is not possible to download BSP files for Planets or Satellites.

Parameters

- **identifier** (*str*, *list*) – Identifier of the object(s). It can be the *name*, *number* or *SPK ID*. It can also be a list of objects.

  Examples: `'2137295'`, `'1999 RB216'`, `'137295'`, `['2137295', '136199', '1999 RC216', 'Chariklo']`.

- **initial_date** (*str*) – Date the bsp file is to begin, within span *1900-2100*.

  Examples: `'2003-02-01'`, `'2003-3-5'`.

- **final_date** (*str*) – Date the bsp file is to end, within span [1900-2100]. Must be more than 32 days later than *initial_date*.

  Examples: `'2006-01-12'`, `'2006-1-12'`.

- **email** (*str*) – User's e-mail contact address. Required by JPL web service.

  Example: `username@user.domain.name`.

- **directory** (*str*) – Directory path to save the bsp files.

6.3 sora.extra

6.3.1 ChiSquare Class

class `sora.extra.`**ChiSquare**(*chi2*, *npts*, ***kwargs*)

Stores the arrays for all inputs and given chi-square.

Parameters

- **chi2** (*array*) – Array with all the chi-square values.
- **npts** (*int*) – Number of points used in the fit.
- ****kwargs** – Any other given input must be an array with the same size as chi2. The keyword *name* will be associated as the variable *name* of the given data.

Example

```
>>> chisquare = ChiSquare(chi2, immersion=t1, emersion=t2)
```

`t1` and `t2` must be an array with the same size as chi2.

The data can be accessed as

```
>>> chisquare.data['immersion']
```

get_nsigma(*sigma=1*, *key=None*)

Determines the interval of the chi-square within the nth sigma.

Parameters

- **sigma** (*float*, *int*) – Value of sigma to calculate.
- **key** (*str*, default=None) – keyword the user desire to obtain results.

Returns
Dictionary with the average n-sigma value and bondaries.

Return type
dict

Note: If a key value is given the mean value within the n-sigma and the error bar within the n-sigma are returned.

If no key is given, a dictionary with: the minimum chi-square, the sigma required, the number of points where $chi2 < (chi2_min + sigma^2)$, and the mean values and errors for all keys is returned.

---

get_values(*sigma=0.0*, *key=None*)

Returns all values where the chi-square is within the nth sigma.

Parameters

- **sigma** (*float*, *int*) – Value of sigma to cut values.
- **key** (*str*) – Keyword (parameter) that the user desires to obtain results of.

Returns
list or dict – List or dictionary with chi-square values within the nth sigma, the average n-sigma value, and bondaries.

Return type
list, *dict*

Note: If a *key* is given, it returns a *list* with all the values that are within the chosen n-sigma.

If no *key* is given, it returns a dictionary with the list with all the values that are within the n-sigma for all keys.

If `sigma=0`, it returns the parameters for the minimum chi-square instead of a list.

---

plot_chi2(*key=None*, *ax=None*)

Plots an ellipse using the input parameters.

Parameters

- **key** (*str*) – Key (parameter) for which to plot the chi squares. If no key is given, it will plot for all parameters.
- **ax** (*maplotlib.pyplot.Axes*) – Matplotlib plot axes. If none is given, it uses the matplotlib pool to identify.

to_file(*namefile*)

Saves the data to a file.

Parameters
namefile (*str*) – Filename to save the data.

to_log(*namefile*)

Saves the chi-squared log to a file.

Parameters
namefile (*str*) – Filename to save the log.

6.3.2 Plot ellipse

sora.extra.plots.**draw_ellipse**(*equatorial_radius*, *oblateness=0.0*, *center_f=0.0*, *center_g=0.0*, *position_angle=0.0*, *center_dot=False*, *ax=None*, ***kwargs*)

Plots an ellipse with the given input parameters.

Parameters

- **equatorial_radius** (*float*, *int*, default=0) – Semi-major axis of the ellipse.
- **oblateness** (*float*, *int*, default=0) – Oblateness of the ellipse.
- **center_f** (*float*, *int*, default=0) – Coordinate of the ellipse (abscissa).
- **center_g** (*float*, *int*, default=0) – Coordinate of the ellipse (ordinate).
- **center_dot** (*bool*, default=False) – If True, plots a dot at the center of the ellipse.
- **position_angle** (*float*, *int*, default= 0) – Pole position angle (Default=0.0).
- **ax** (*maptlotlib.pyplot.Axes*) – Axis where to plot ellipse.
- ****kwargs** – All other parameters. They will be parsed directly by matplotlib.

6.3.3 Complementary functions

sora.extra.utils.**get_ellipse_points**(*theta*, *equatorial_radius*, *oblateness=0.0*, *center_f=0.0*, *center_g=0.0*, *position_angle=0.0*)

Get points for the ellipse with the given input parameters.

Parameters

- **theta** (*float* array) – Angular coordinate, in degrees, to return the ellipse points.
- **equatorial_radius** (*float*, *int*) – Semi-major axis of the ellipse.
- **oblateness** (*float*, *int*, default=0) – Oblateness of the ellipse.
- **center_f** (*float*, *int*, default=0) – Coordinate of the ellipse (abscissa).
- **center_g** (*float*, *int*, default=0) – Coordinate of the ellipse (ordinate).

- **position_angle** (*float*, *int*, default=0) – The pole position angle of the ellipse in degrees. Zero is in the North direction ('g-positive'). Positive clockwise.

Returns

- **x_model** (*float*, array) – Cartesian x-component, in km
- **y_model** (*float*, array) – Cartesian y-component, in km
- **r_model** (*float*, array) – Radial distance, in km
- **theta** (*float* array) – Angular coordinate, in degrees, to return the ellipse points.

6.4 sora.lightcurve

6.4.1 The LightCurve Class

class sora.**LightCurve**(*name=''*, ***kwargs*)

Defines a Light Curve.

Parameters

- **name** (*str*) – The name of the LightCurve. Each time an LightCurve object is defined the name must be different.
- **tref** (*astropy.time.Time*, *str*, *float*) – Instant of reference.

  Format: *Julian Date*, string in ISO format or Time object. Required only if LightCurve have input fluxes and given time is not in Julian Date.
- **central_bandpass** (*int*, *float*, otpional, default=0.7) – The center band pass of the detector used in observation. Value in microns.
- **delta_bandpass** (*int*, *float*, optional, default=0.3) – The band pass width of the detector used in observation. Value in microns.
- **exptime** (*int*, *float*) – The exposure time of the observation, in seconds. *NOT* required in cases *2*, *3* and *4* below. *Required* in case *1* below.
- ****kwargs** (*int*, *float*) – Object velocity, distance, and star diameter.

Note:

vel
[*int*, *float*] Velocity in km/s.

dist
[*int*, *float*] Object distance in AU.

d_star
[*float*] Star diameter, in km.

Warning: Input data must be one of the 4 options below:

1) **Input data from file with time and flux**
file (str): a file with the time and flux. A third column with the error in flux can also be given.

usecols (int, tuple, array): Which columns to read, with the first being the time, the seconds the flux and third the flux error (optional).

Example:

```
>>> LightCurve(name, file, exptime) # dflux can also be given
```

2) **Input data when file is not given:**

time: time must be a list of times, in seconds from tref, or Julian Date, or a Time object.

flux: flux must be a list of fluxes. It must have the same lenght as time.

dflux: if file not given, dflux must be a list of fluxes errors. It must have the same lenght as time. (not required)

Example:

```
>>> LightCurve(name, flux, time, exptime) # dflux can also be given
```

Cases for when *time* and *flux* are not given.

3) **Input for a positive occultation:**

immersion: The instant of immersion.

emersion: The instant of emersion.

immersion_err: Immersion time uncertainty, in seconds.

emersion_err: Emersion time uncertainty, in seconds.

Example:

```
>>> LightCurve(name, immersion, immersion_err, emersion, emersion_err)
```

4) **Input for a negative occultation:**

initial_time: The initial time of observation.

end_time: The end time of observation.

Example:

```
>>> LightCurve(name, initial_time, end_time)
```

`calc_magnitude_drop`(*mag_star*, *mag_obj*)

Determines the magnitude drop of the occultation.

Parameters

- **`mag_star`** (*int*, *float*) – Star magnitude.
- **`int`** (*mag_obj*) – Object apparent magnitude to the date.
- **`float`** – Object apparent magnitude to the date.

Returns

- **mag_drop** (*float*) – Magnitude drop for the given magnitudes.
- **bottom_flux** (*float*) – Normalized bottom flux for the given magnitudes.

normalize(*poly_deg=None*, *mask=None*, *flux_min=0.0*, *flux_max=1.0*, *plot=False*)

Returns the normalized flux within the flux min and flux max defined scale.

Parameters

- **poly_deg** (*int*) – Degree of the polynomial to be fitted.
- **mask** (*bool* array) – Which values to be fitted.
- **flux_min** (*int*, *float*) – Event flux to be set as 0.
- **flux_max** (*int*, *float*) – Baseline flux to be set as 1.
- **plot** (*bool*) – If True plot the steps for visual aid.

occ_detect(*maximum_duration=None*, *dur_step=None*, *snr_limit=None*, *n_detections=None*, *plot=False*)

Detects automatically the occultation event in the light curve.

Detects a 'square well' shaped transit. All parameters are optional.

Parameters

- **maximum_duration** (*float*, default: light curve time span) – Maximum duration of the occultation event.
- **dur_step** (*float*, default: 1/2 of sampling rate) – Step size to sweep occultation duration event.
- **snr_limit** (*float*, default=None) – Minimum occultation SNR.
- **n_detections** (*int*, default=1) – Number of detections regardless the SNR. *n_detections* is superseded by *snr_limit*.
- **plot** (*bool*) – True if output plots are desired.

Returns

OrderedDict – An ordered dictionary of `name`:`value` pairs for each parameter.

Return type

dict

Examples

```
>>> lc = LightCurve(time=time, flux=flux, exptime=0.0, name='lc_example')
>>> params = lc.occ_detect()
>>> params
{'rank': 1,
'occultation_duration': 40.1384063065052,
'central_time': 7916.773870512843,
'immersion_time': 7896.7046673595905,
'emersion_time': 7936.843073666096,
'time_err': 0.05011036992073059,
'depth': 0.8663887801707082,
'depth_err': 0.10986223384336465,
'baseline': 0.9110181732552853,
'baseline_err': 0.19045768512595365,
'snr': 7.886138392251848,
'occ_mask': array([False, False, False, ..., False, False, False])}
```

`occ_lcfit`(***kwargs*)

Monte Carlo chi square fit for occultations lightcurve.

Parameters

- **tmin** (*int*, *float*) – Minimum time to consider in the fit procedure, in seconds.
- **tmax** (*int*, *float*) – Maximum time to consider in the fit procedure, in seconds.
- **flux_min** (*int*, *float*, default=0) – Bottom flux (only object).
- **flux_max** (*int*, *float*, default=1) – Base flux (object plus star).
- **immersion_time** (*int*, *float*) – Initial guess for immersion time, in seconds.
- **emersion_time** (*int*, *float*) – Initial guess for emersion time, in seconds.
- **opacity** (*int*, *float*, default=1) – Initial guess for opacity. Opaque = 1, Transparent = 0.
- **delta_t** (*int*, *float*) – Interval to fit immersion or emersion time.
- **dopacity** (*int*, *float*, default=0) – Interval to fit opacity.
- **sigma** (*int*, *float*, *array*, 'auto') – Fluxes errors. If None it will use the *self.dflux*. If 'auto' it will calculate using the region outside the event.
- **loop** (*int*, default=10000) – Number of tests to be done.
- **verbose** (*bool*, default=False) – If True, it prints information while fitting.
- **sigma_result** (*int*, *float*) – Sigma value to be considered as result.
- **method** (*str*, default=`chisqr`) – Method used to perform the fit. Available methods are: *chisqr* : monte carlo computation method used in versions of SORA <= 0.2.1. *fastchi* : monte carlo computation method, allows multithreading. *least_squares* or *ls*: best fit done used levenberg marquardt convergence algorithm. *differential_evolution* or *de*: best fit done using genetic algorithms. All methods return a Chisquare object.
- **threads** (*int*) – Number of threads/workers used to perform parallel computations of the chi square object. It works with all methods except *chisqr*, by default 1.

Returns
chi2 – ChiSquare object.

Return type
sora.extra.ChiSquare

`occ_model`(*immersion_time*, *emersion_time*, *opacity*, *mask*, *npt_star=12*, *time_resolution_factor=10*, *flux_min=0*, *flux_max=1*)

Returns the modelled light curve.

The modelled light curve takes into account the fresnel diffraction, the star diameter and the instrumental response.

Parameters

- **immersion_time** (*int*, *float*) – Immersion time, in seconds.
- **emersion_time** (*int*, *float*) – Emersion time, in seconds.
- **opacity** (*int*, *float*) – Opacity. Opaque = 1.0, transparent = 0.0,
- **mask** (*bool* array) – Mask with True values to be computed.
- **npt_star** (*int*, default=12) – Number of subdivisions for computing the star size effects.

- **time_resolution_factor** (*int*, *float*, default: 10*fresnel scale) – Steps for fresnel scale used for modelling the light curve.
- **flux_min** (*int*, *float*, default=0) – Bottom flux (only object).
- **flux_max** (*int*, *float*, default=1) – Base flux (object plus star).

plot_lc(*ax=None*)

Plots the light curve

plot_model(*ax=None*)

Plots the modelled light curve

reset_flux()

Resets flux for original values

set_dist(*dist*)

Sets the object distance.

Parameters
dist (*int*, *float*) – Object distance in AU.

set_exptime(*exptime*)

Sets the light curve exposure time.

Parameters
exptime (*int*, *float*) – Exposure time, in seconds.

set_filter(*central_bandpass*, *delta_bandpass*)

Sets the filter bandwidth in microns.

Parameters

- **central_bandpass** (*float*) – Center band in microns.
- **delta_bandpass** (*float*) – Bandwidth in microns.

set_flux(***kwargs*)

Sets the flux for the LightCurve.

Parameters

- **exptime** (*int*, *float*, required) – The exposure time of the observation, in seconds.
- **file** (*str*) – A file with the time and flux in the first and second columns, respectively. A third column with error in flux can also be given.
- **time** – If file not given, time must be a list of times, in seconds from *tref*, or *Julian Date*, or a *Time object*.
- **flux** – If file not given, flux must be a list of fluxes. It must have the same lenght as time.
- **dflux** – If file not given, dflux must be a list of fluxes errors. It must have the same lenght as time.
- **tref** (*astropy.time.Time*, *str*, *float*) – Instant of reference. It can be in *Julian Date*, string in ISO format or *Time object*.
- **usecols** (*int*, *tuple*, array, optional) – Which columns to read, with the first being the time, the seconds the flux and third the flux error.
- ****kwargs** (*int*, *float*) – Object velocity, object distance, star diameter.

Note:

vel
[*int*, *float*] Velocity in km/s.

dist
[*int*, *float*:] Object distance in AU.

d_star
[*float*] Star diameter, in km.

`set_star_diam`(*d_star*)

Sets the star diameter.

Parameters
d_star (*float*) – Star diameter, in km.

`set_vel`(*vel*)

Sets the occultation velocity.

Parameters
vel (*int*, *float*) – Velocity in km/s.

`to_file`(*namefile=None*)

Saves the light curve to a file.

Parameters
namefile (*str*) – Filename to save the data.

`to_log`(*namefile=None*)

Saves the light curve log to a file.

Parameters
namefile (*str*) – Filename to save the log.

6.4.2 Occutation Detection

`sora.lightcurve.occdetect.occ_detect`(*flux*, *dflux*, *time*, *cycle*, *maximum_duration=None*, *dur_step=None*, *snr_limit=None*, *n_detections=None*, *plot=False*)

Detects automatically the occultation event in the light curve.

Detects a 'square well' shaped transit. All parameters are optional.

Parameters

- **flux** (*float* array) – Flux of the time series. Dependent variable.
- **dflux** (*float* array) – Error in the flux. Error in the dependent variable.
- **time** (*float* array) – Time variable. Independent variable.
- **cycle** (*float*) – Sampling value of the time series.
- **maximum_duration** (*float*, default: light curve time span) – Maximum duration of the occultation event.
- **dur_step** (*float*, default: 1/2 cycle) – Step size to sweep occultation duration event.
- **snr_limit** (*float*, default=None) – Minimum occultation SNR.
- **n_detections** (*int*, default=1) – Number of detections regardless the SNR. *n_detections* is superseded by *snr_limit*.

- **plot** (*boolean*, default=False) – True if output plots are desired.

Returns
OrderedDict – An ordered dictionary of `name`:`value` pairs for each parameter.

Return type
dict

Examples

```
>>> from sora.lightcurve.occdetect import occ_detect
>>> params = occ_detect(flux, dflux, time, 0.2)
>>> params
{'rank': 1,
'occultation_duration': 40.1384063065052,
'central_time': 7916.773870512843,
'immersion_time': 7896.7046673595905,
'emersion_time': 7936.843073666096,
'time_err': 0.05011036992073059,
'depth': 0.8663887801707082,
'depth_err': 0.10986223384336465,
'baseline': 0.9110181732552853,
'baseline_err': 0.19045768512595365,
'snr': 7.886138392251848,
'occ_mask': array([False, False, False, ..., False, False, False])}
```

6.4.3 Complementary functions

`sora.lightcurve.utils.`**`calc_fresnel`**(*distance*, *bandpass*)
Calculates the Fresnel scale.

Fresnel Scale = square root of half the multiplication of wavelength and object distance.

Parameters

- **distance** (*int*, *float* array) – Distances, in km.
- **bandpass** (*int*, *float*, array) – Wavelength, in km.

Returns
fresnel_scale – Fresnel scale, in km.

Return type
float, array

`sora.lightcurve.utils.`**`calc_magnitude_drop`**(*mag_star*, *mag_obj*)
Determines the magnitude drop of the occultation.

Parameters

- **mag_star** (*int*, *float*) – Star magnitude.
- **mag_obj** (*int*, *float*) – Object apparent magnitude to the date.

Returns

- **mag_drop** (*float*) – Magnitude drop for the given magnitudes.
- **bottom_flux** (*float*) – Normalized bottom flux for the given magnitudes.

6.5 sora.observer

6.5.1 The Observer Class

class sora.**Observer**(***kwargs*)

Defines the observer object.

name

Name for the Observer. Observer is uniquely defined (name must be different for each observer).

Type
str

code

The IAU code for SORA to search for its coordinates in MPC database.

Type
str

site

User provides an EarthLocation object.

Type
astropy.coordinates.EarthLocation

lon

The Longitude of the site in degrees. Positive to East. Range (0 to 360) or (-180 to +180). User can provide in degrees (*float*) or hexadecimal (*string*).

Type
str, *float*

lat

The Latitude of the site in degrees. Positive North. Range (+90 to -90). User can provide in degrees (float) or hexadecimal (string).

Type
str, *float*

height

The height of the site in meters above see level.

Type
int, *float*

ephem

The ephemeris used to locate the observer on space. It can be "horizons" to use horizons or a list of kernels

Type
str, *list*

Examples

User can provide one of the following to define an observer:

- If user will use the MPC name for the site:

```
>>> Observer(code)
```

- If user wants to use a different name from the MPC database:

```
>>> Observer(name, code)
```

- If user wants to use an EarthLocation value:

```
>>> from astropy.coordinates import EarthLocation
>>> EarthLocation(lon, lat, height)
>>> Observer(name, site)
```

- If user wants to give site coordinates directly:

```
>>> Observer(name, lon, lat, height)
```

altaz(*time*, *coord*)

Calculates the Altitude and Azimuth at a reference time for a coordinate.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the sidereal time. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **coord** (*str*, *astropy.coordinates.SkyCoord*) – Coordinate of the target ICRS.

Returns

- **altitude** (*float*) – Object altitude in degrees.
- **azimuth** (*float*) – Object azimuth in degrees.

get_ksi_eta(*time*, *star*)

Calculates relative position to star in the orthographic projection.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the position. It can be a string in the format `'yyyy-mm-dd hh:mm:ss.s'` or an astropy *Time* object.
- **star** (*str*, *astropy.coordinates.SkyCoord*) – The coordinate of the star in the same reference frame as the ephemeris. It can be a string in the format `'hh mm ss.s +dd mm ss.ss'` or an astropy *SkyCoord* object.

Returns

ksi, eta – On-sky orthographic projection of the observer relative to a star. `ksi` is in the North-South direction (North positive). `eta` is in the East-West direction (East positive).

Return type

float

`get_vector`(*time*, *origin='barycenter'*)

Return the vector Origin -> Observer in the ICRS

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the object position. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **origin** (*str*) – Origin of vector. It can be 'barycenter' or 'geocenter'.

Returns
coord – Astropy SkyCoord object with the vector origin -> observer at the given time.

Return type
astropy.coordinates.SkyCoord

`sidereal_time`(*time*, *mode='local'*)

Calculates the Apparent Sidereal Time at a reference time.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate sidereal time.It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **mode** (*str*) – Local or greenwich time. If mode set `'local'` calculates the sidereal time for the coordinates of this object. If mode set `'greenwich'` calculates the Greenwich Apparent Sidereal Time.

Returns
An Astropy Longitude object with the Sidereal Time.

Return type
sidereal_time

`to_log`(*namefile*)

Saves the observer log to a file.

Parameters
namefile (*str*) – Filename to save the log.

6.5.2 The Spacecraft Class

`class sora.Spacecraft`(*name*, *spkid*, *ephem='horizons'*)

Defines a spacecraft observer object.

`name`

Name for the Observer. Observer is uniquely defined (name must be different for each observer).

Type
str

`spkid`

spkid of the targeting object.

Type
str, required

`ephem`

The ephemeris used to locate the observer on space. It can be "horizons" to use horizons or a list of kernels

Type
str, *list*

`get_vector`(*time*, *origin='barycenter'*)

Return the vector Origin -> Observer in the ICRS

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the object position. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **origin** (*str*) – Origin of vector. It can be 'barycenter' or 'geocenter'.

Returns
coord – Astropy SkyCoord object with the vector origin -> observer at the given time.

Return type
astropy.coordinates.SkyCoord

`to_log`(*namefile*)

Saves the spacecraft log to a file.

Parameters
namefile (*str*) – Filename to save the log.

6.5.3 Complementary functions

sora.observer.utils.**search_code_mpc**()

Reads the MPC Observer Database.

Returns
observatories – A python dictionary with all the sites as Astropy EarthLocation objects.

Return type
dict

6.6 sora.occultation

6.6.1 The Occultation Class

class sora.**Occultation**(*star*, *body=None*, *ephem=None*, *time=None*, *reference_center='geocenter'*)

Instantiates the Occultation object and performs the reduction of the occultation.

star

the coordinate of the star in the same reference frame as the ephemeris. It must be a Star object or a string with the coordinates of the object to search on Vizier.

Type
sora.Star, *str*, required

body

Object that will occult the star. It must be a Body object or its name to search in the Small Body Database.

Type
sora.Body, *str*

ephem

Object ephemeris. It must be an Ephemeris object or a list.

Type
sora.Ephem, *list*

time

Reference time of the occultation. Time does not need to be exact, but needs to be within approximately 50 minutes of the occultation closest approach to calculate occultation parameters.

Type
str, *astropy.time.Time*, required

reference_center

A SORA observer object or a string 'geocenter'. The occultation parameters will be calculated in respect to this reference as center of projection.

Type
str, *sora.Observer*, *sora.Spacecraft*

Important: When instantiating with "body" and "ephem", the user may define the Occultation in 3 ways:

1. With *body* and *ephem*.

2. With only "body". In this case, the "body" parameter must be a Body object and have an ephemeris associated (see Body documentation).

3. With only *ephem*. In this case, the *ephem* parameter must be one of the Ephem Classes and have a name (see Ephem documentation) to search for the body in the Small Body Database.

---

add_observation(*obs*, *lightcurve*)

Adds observations to the Occultation object.

Parameters

- **obs** (*sora.Observer*) – The Observer object to be added.
- **lightcurve** (*sora.LightCurve*) – The LightCurve object to be added.

check_time_shift(*time_interval=30*, *time_resolution=0.001*, *verbose=False*, *plot=False*, *use_error=True*, *delta_plot=100*, *ignore_chords=None*)

Check the needed time offset, so all chords have their center aligned.

Parameters

- **time_interval** (*int*, *float*) – Time interval to check, default is 30 seconds.
- **time_resolution** (*int*, *float*) – Time resolution of the search, default is 0.001 seconds.
- **verbose** (*bool*) – If True, it prints text, default is False.
- **plot** (*bool*, default=False) – If True, it plots figures as a visual aid.
- **use_error** (*bool*, default=True) – if True, the linear fit considers the time uncertainty.
- **delta_plot** (*int*, *float*, default=100) – Value to be added to increase the plot limit, in km.
- **ignore_chords** (*str*, list, default=None) – Names of the chords to be ignored in the linear fit.

Returns
time_shift – Dictionary with needed time offset to align the chords, each key is the name of the chord.

Return type
dict

`check_velocities`()

Prints the current velocity used by the LightCurves and its radial velocity.

`fit_ellipse`(***kwargs*)

Fits an ellipse to given occultation using given parameters.

Parameters

- **center_f** (*int*, *float*, default=0) – The coordinate in f of the ellipse center.
- **center_g** (*int*, *float*, default=0) – The coordinate in g of the ellipse center.
- **equatorial_radius** (*int*, *float*) – The Equatorial radius (semi-major axis) of the ellipse.
- **oblateness** (*int*, *float*, default=0) – The oblateness of the ellipse.
- **position_angle** (*int*, *float*, default=0) – The pole position angle of the ellipse in degrees. Zero is in the North direction ('g-positive'). Positive clockwise.
- **dcenter_f** (*int*, *float*) – Interval for coordinate f of the ellipse center.
- **dcenter_g** (*int*, *float*) – Interval for coordinate g of the ellipse center.
- **int** (*dequatorial_radius*) – Interval for the Equatorial radius (semi-major axis) of the ellipse.
- **float** – Interval for the Equatorial radius (semi-major axis) of the ellipse.
- **doblateness** (*int*, *float*) – Interval for the oblateness of the ellipse
- **dposition_angle** (*int*, *float*) – Interval for the pole position angle of the ellipse in degrees.
- **loop** (*int*, default=10000000) – The number of ellipses to attempt fitting.
- **dchi_min** (*int*, *float*) – If given, it will only save ellipsis which chi square are smaller than chi_min + dchi_min. By default *None* when used with *method='chisqr'*, and *3* for other methods.
- **number_chi** (*int*, default=10000) – In the *chisqr* method if *dchi_min* is given, the procedure is repeated until *number_chi* is reached. In other methods it is the number of values (simulations) that should lie within the provided *sigma_result*.
- **verbose** (*bool*, default=True) – If True, it prints information while fitting.
- **ellipse_error** (*int*, *float*) – Model uncertainty to be considered in the fit, in km.
- **sigma_result** (*int*, *float*) – Sigma value to be considered as result.
- **method** (*str*, *default='least_squares'*) – Method used to perform the fit. Available methods are: *chisqr* : monte carlo computation method used in versions of SORA <= 0.2.1. *fastchi* : monte carlo computation method, allows multithreading . *least_squares* or *ls*: best fit done used levenberg marquardt convergence algorithm. *differential_evolution* or *de*: best fit done using genetic algorithms. All methods return a Chisquare object.
- **threads** (*int*) – Number of threads/workers used to perform parallel computations of the chi square object. It works with all methods except *chisqr*, by default 1.

Returns
chisquare – A ChiSquare object with all parameters.

Return type
sora.ChiSquare

Important: Each occultation is added as the first argument(s) directly.

Mandatory input parameters: 'center_f', 'center_g', 'equatorial_radius', 'oblateness', and 'position_angle'.

Parameters fitting interval: 'dcenter_f', 'dcenter_g', 'dequatorial_radius', 'doblateness', and 'dposition_angle'. Default values are set to zero. Search done between (value - dvalue) and (value + dvalue).

Examples

To fit the ellipse to the chords of occ1 Occultation object:

```
>>> fit_ellipse(occ1, **kwargs)
```

To fit the ellipse to the chords of occ1 and occ2 Occultation objects together:

```
>>> fit_ellipse(occ1, occ2, **kwargs)
```

`fit_shape`(***kwargs*)

Parameters

- **occ** (*sora.Occultation*) – The Occultation object with information to fit
- **center_f** (*int*, *float*, default=0) – The coordinate in f of the ellipse center.
- **center_g** (*int*, *float*, default=0) – The coordinate in g of the ellipse center.
- **dcenter_f** (*int*, *float*) – Interval for coordinate f of the ellipse center.
- **dcenter_g** (*int*, *float*) – Interval for coordinate g of the ellipse center.
- **scale** (*number*) – Scale factor of the limb
- **dscale** (*number*) – Interval for scale
- **loop** (*int*, default=150000) – The number of centers to attempt fitting.
- **sigma_result** (*int*, *float*) – Sigma value to be considered as result.

Returns
chisquare – A ChiSquare object with all parameters.

Return type
sora.ChiSquare

`get_map_sites`()

Returns Dictionary with sites in the format required by plot_occ_map function.

Returns
sites – Dictionary with the sites in the format required by *plot_occ_map* function.

Return type
dict

new_astrometric_position(*time=None*, *offset=None*, *error=None*, *verbose=True*, *observer=None*)

Calculates the new astrometric position for the object given fitted parameters.

Parameters

- **time** (*str*, *astropy.time.Time*) – Reference time to calculate the position. If not given, it uses the instant of the occultation Closest Approach. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **offset** (*list*) – Offset to apply to the position. If not given, uses the parameters from the fitted ellipse.

  Note: Must be a list of 3 values being [X, Y, 'unit']. 'unit' must be Angular or Distance unit.

  If Distance units for X and Y: Ex: [100, -200, 'km'], [0.001, 0.002, 'AU']

  If Angular units fox X [d*a*cos(dec)] and Y [d*dec]: Ex: [30.6, 20, 'mas'], or [-15, 2, 'arcsec']

- **error** (*list*) – Error bar of the given offset. If not given, it uses the 1-sigma value of the fitted ellipse.

  Note: Error must be a list of 3 values being [dX, dY, 'unit'], similar to offset. It does not need to be in the same unit as offset.

- **verbose** (*bool*) – If true, it Prints text, else it Returns text.
- **observer** (*str*, *sora.Observer*, *sora.Spacecraft*) – IAU code of the observer (must be present in given list of kernels), a SORA observer object or a string: ['geocenter', 'barycenter']

observations()

Print all the observations added to the Occultation object Pair (*Observer*, *LightCurve*)

plot_chords(*all_chords=True*, *positive_color='blue'*, *negative_color='green'*, *error_color='red'*, *ax=None*, *lw=2*)

Plots the chords of the occultation.

Parameters

- **all_chords** (*bool*, default=True) – If True, it plots all the chords. If False, it sees what was deactivated in self.positions and ignores them.
- **positive_color** (*str*, default='blue') – Color for the positive chords.
- **negative_color** (*str*, default='green') – Color for the negative chords.
- **error_color** (*str*, default='red') – Color for the error bars of the chords.
- **ax** (*maptlotlib.pyplot.Axes*, default=None) – Axis where to plot chords (default: Uses matplotlib pool).
- **lw** (*int*, *float*, default=2) – Linewidth of the chords.

plot_occ_map(***kwargs*)

Parameters

- **radius** (*int*, *float*) – The radius of the shadow. If not given it uses saved value.

- **nameimg** (*str*) – Change the name of the image saved.
- **path** (*str*) – Path to a directory where to save map.
- **resolution** (*int*, default=2) –
- **resolution.** (*Cartopy feature*) –
- **"10m";** (*- 1 means a resolution of*) –
- **"50m";** (*- 2 a resolution of*) –
- **"100m".** (*- 3 a resolution of*) –
- **states** (*bool*) – If True, plots the states borders of the countries. The states of some countries will only be shown depending on the resolution.
- **zoom** (*int*, *float*) – Zooms in or out of the map.
- **centermap_geo** (*list*, default=None) – Center the map given coordinates in longitude and latitude. It must be a list with two numbers.
- **centermap_delta** (*list*, default=None) – Displace the center of the map given displacement in X and Y, in km. It must be a list with two numbers.
- **centerproj** (*list*) – Rotates the Earth to show occultation with the center projected at a given longitude and latitude. It must be a list with two numbers.
- **labels** (*bool*, default=True) – Plots text above and below the map with the occultation parameters.
- **meridians** (*int*, default=30) – Plots lines representing the meridians for given interval, in degrees.
- **parallels** (*int*, default=30) – Plots lines representing the parallels for given interval, in degrees.
- **sites** (*dict*) – Plots site positions in map. It must be a python dictionary where the key is the *name* of the site, and the value is a list with *longitude*, *latitude*, *delta_x*, *delta_y* and *color*. *delta_x* and *delta_y* are displacement, in km, from the point position of the site in the map and the *name*. *color* is the color of the point.
- **site_name** (*bool*) – If True, it prints the name of the sites given, else it plots only the points.
- **countries** (*dict*) – Plots the names of countries. It must be a python dictionary where the key is the name of the country and the value is a list with longitude and latitude of the lower left part of the text.
- **offset** (*list*) – Applies an offset to the ephemeris, calculating new CA and instant of CA. It is a pair of *delta_RA*cosDEC* and *delta_DEC*.
- **mapstyle** (*int*, default=1) – Define the color style of the map. `'1'` is the default black and white scale. `'2'` is a colored map.
- **error** (*int*, *float*) – Ephemeris error in mas. It plots a dashed line representing radius + error.
- **ercolor** (*str*) – Changes the color of the lines of the error bar.
- **ring** (*int*, *float*) – Plots a dashed line representing the location of a ring. It is given in km, from the center.
- **rncolor** (*str*) – Changes the color of ring lines.

- **atm** (*int*, *float*) – Plots a dashed line representing the location of an atmosphere. It is given in km, from the center.
- **atcolor** (*str*) – Changes the color of atm lines.
- **chord_delta** (*list*) – List with distances from center to plot chords.
- **chord_geo** (*2d-list*) – List with pairs of coordinates to plot chords.
- **chcolor** (*str*, default='grey') – Color of the line of the chords.
- **heights** (*list*) – It plots a circular dashed line showing the locations where the observer would observe the occultation at a given height above the horizons. This must be a list.
- **hcolor** (*str*) – Changes the color of the height lines.
- **mapsize** (*list*, default= [46.0, 38.0]) – The size of figure, in cm. It must be a list with two values.
- **cpoints** (*int*, *float*, default=60) – Interval for the small points marking the center of shadow, in seconds.
- **ptcolor** (*str*) – Change the color of the center points.
- **alpha** (*float*, default=0.2) – The transparency of the night shade, where 0.0 is full transparency and 1.0 is full black.
- **fmt** (*str*, default:'png') – The format to save the image. It is parsed directly by *matplotlib.pyplot*.
- **dpi** (*int*, default=100) – Resolution in "dots per inch". It defines the quality of the image.
- **lncolor** (*str*) – Changes the color of the line that represents the limits of the shadow over Earth.
- **outcolor** (*str*) – Changes the color of the lines that represents the limits of the shadow outside Earth.
- **scale** (*int*, *float*) – Arbitrary scale for the size of the name of the site.
- **cscale** (*int*, *float*) – Arbitrary scale for the name of the country.
- **sscale** (*int*, *float*) – Arbitrary scale for the size of point of the site.
- **pscale** (*int*, *float*) – Arbitrary scale for the size of the points that represent the center of the shadow.
- **arrow** (*bool*) – If True, it plots the arrow with the occultation direction.

Note: Only one of centermap_geo and centermap_delta can be given.

---

plot_radial_dispersion(*ax=None*, ***kwargs*)

Plots the radial dispersion

A shape must have been fitted to the chords

Parameters

- **ax** (*matplotlib.pyplot.Axes*) – The axes where to make the plot. If None, it will use the default axes.
- ****kwargs** – Any other kwarg will be parsed directly by *maplotlip.pyplot.plot*. The only difference is that the default linewidth *lw=1* and marker is *marker=o*.

property positions

Calculates the position and velocity for all chords. Saves it into an *_PositionDict* object.

remove_observation(*key*, *key_lc=None*)

Removes an observation from the Occultation object.

Parameters

- **key** (*str*) – The name given to *Observer* or *LightCurve* to remove from the list.
- **key_lc** (*str*) – In the case where repeated names are present for different observations, *key_lc* must be given for the name of the *LightCurve* and key will be used for the name of the *Observer*.

to_file()

Saves the occultation data to a file.

Three files are saved containing the positions and velocities for the observations. They are for the positive, negative and error bars positions.

The format of the files are: positions in f and g, velocities in f and g, the Julian Date of the observation, light curve name of the corresponding position.

to_log(*namefile=None*)

Saves the occultation log to a file.

Parameters

namefile (*str*) – Filename to save the log.

6.6.2 The Chord Class

class sora.occultation.**Chord**(*, *name*, *observer*, *lightcurve*)

Defines an Occultation Chord.

name

The name of the Chord.

Type

str

observer

The site of observation.

Type

sora.Observer

lightcurve

The lightcurve observed.

Type

sora.LightCurve

disable(*, *time=None*)

Disables a contact point of the curve to be used in the fit.

Parameters

time (*None*, *str*) – If None, it will disable all contact points. If `'immersion'` or `'emersion'`, it will disable respective contact point.

enable(*, *time=None*)

Enables a contact point of the curve to be used in the fit.

Parameters

time (*None*, *str*) – If `None`, it will enable all contact points. If `'immersion'` or `'emersion'`, it will enable respective contact point.

get_fg(*, *time=None*, *vel=False*)

Returns the on-sky position for this chord at a given time.

This Chord object must be associated to an Occultation to work, since it needs the position of the star and an ephemeris.

Parameters

- **time** (*astropy.time.Time*, *str*) – It must a time or a list of time to calculate the on-sky position. If time is `'immersion'`, `'emersion'`, `'start'` or `'end'`, it will get the respective on-sky position. If `time=None`, it will return the on-sky position for the immersion and emersion times if positive or raise an error if negative.
- **vel** (*bool*) – If True, it will return the on-sky velocity as well.

Returns

f, g, [vf, vg] – The geocentric on-sky Orthographic projection of the object with *f* pointing to the celestial North and *g* pointing to the celestial East. The respective velocities (vf, vg) are returned if `vel=True`.

Return type

float, *float*, [*float*, *float*]

get_impact_param(*center_f=0*, *center_g=0*, *verbose=True*)

Gets the impact parameter, minimal distance between the chord and the centre position.

This Chord object must be associated to an Occultation to work, since it needs the position of the star and an ephemeris.

Parameters

- **center_f** (*int*, *float*, default=0) – The coordinate in f of the ellipse center;
- **center_g** (*int*, *float*, default=0) – The coordinate in g of the ellipse center.
- **verbose** (*bool*, default=True) – If True, prints the obtained values.

Returns

impact, sense – The impact parameter (in km) and the direction of the chord relative the ellipse center, North (N), South (S), East (E) and West (W).

Return type

list

get_limb_points(*only_able=True*)

Computes the projected points and errors on the tangent plane

Parameters

only_able (*bool*) – Get only the contact points that are able to be used in the fit.

Returns

- **fg** (*numpy.array*) – The projected points of occultation instants. Each line is a point on the projection with x and y respectively.
- **error** (*numpy.array*) – Error vector of the projected occultation instants. Each line is the error of a point in x and y, respectively.

`get_theoretical_times`(*equatorial_radius*, *center_f=0*, *center_g=0*, *oblateness=0*, *position_angle=0*, *sigma=0*, *step=1*, *verbose=True*)

Gets the theoretical times and chord size for a given ellipse.

This Chord object must be associated to an Occultation to work, since it needs the position of the star and an ephemeris.

Parameters

- **equatorial_radius** (*int*, *float*) – The Equatorial radius (semi-major axis) of the ellipse.
- **center_f** (*int*, *float*, default=0) – The coordinate in f of the ellipse center
- **center_g** (*int*, *float*, default=0) – The coordinate in g of the ellipse center
- **oblateness** (*int*, *float*, default=0) – The oblateness of the ellipse.
- **position_angle** (*int*, *float*, default=0) – The pole position angle of the ellipse in degrees. Zero is in the North direction ('g-positive'). Positive clockwise.
- **sigma** (*int*, *float*) – Uncertainty of the expected ellipse, in km.
- **step** (*int*, *float*) – Time resolution of the chord, in seconds.
- **verbose** (*bool*, default=True) – If True, prints the obtained values.

Returns

theory_immersion_time, theory_emersion_time, theory_chord_size – The expected immersion time for the given ellipse, the expected emersion time for the given ellipse, and the expected chord size for the given ellipse.

Return type

list

`property is_able`

Return a dictionary with the references enabled and disabled for the fit.

`path`(*, *segment='standard'*, *step=1*)

Returns the on-sky path of this chord.

This Chord object must be associated to an Occultation to work, since it needs the position of the star and an ephemeris.

Parameters

- **segment** (*str*) – The segment to get the path of the chord. The available options are:

  `'positive'`: to get the path between the immersion and emersion times if the chord is positive.

  `'negative'`: to get the path between the start and end of observation if the chord is negative.

  `'standard'`: to get the 'positive' path if the chord is positive or 'negative' if the chord is negative.

  `'full'`: to get the path between the start and end of observation independent if the chord is positive or negative.

  `'outer'`: to get the path outside the 'positive' path, for instance between the start and immersion times and between the emersion and end times.

  `'error'`: to get the path corresponding to the error bars. Be aware that some of these segments may return more than 1 path, for example `segment='error'`.

- **step** (*int*, *float*, *str*) – If a number, it corresponds to the step, in seconds, for each point of the path.

  This correspond to an approximate value if it is not a multiple of the interval of the segment.

  The step can also be equal to `'exposure'`. In this case, the path will return a set of pairs where each pair will have the position at the beginning of the exposure and the end of the exposure.

Returns
path f, path g – It will return the path separated by f and g. If `step='exposure'`, it will return f and g intercalated for each exposure.

Return type
array, *array*

plot_chord(*, *segment='standard'*, *only_able=False*, *ax=None*, *linestyle='-'*, *time_direction=False*, ***kwargs*)

Plots the on-sky path of this chord.

This Chord object must be associated to an Occultation to work, since it needs the position of the star and an ephemeris.

Parameters

- **segment** (*str*) – The segment to plot the chord. The available options are:

  `'positive'`: to get the path between the immersion and emersion times if the chord is positive.

  `'negative'`: to get the path between the start and end of observation if the chord is negative.

  `'standard'`: to get the 'positive' path if the chord is positive
  or 'negative' if the chord is negative.

  `'full'`: to get the path between the start and end of observation independent if the chord is positive or negative.

  `'outer'`: to get the path outside the 'positive' path, for instance between the start and immersion times and between the emersion and end times.

  `'error'`: to get the path corresponding to the error bars.

- **only_able** (*bool*) – Plot only the contact points that are able to be used in the fit. If `segment='error'` it will show only the contact points able. If segment is any other, the path will be plotted only if both immersion and emersion are able, or it is a negative chord.
- **ax** (*matplotlib.pyplot.Axes*) – The axes where to make the plot. If None, it will use the default axes.
- **linestyle** (*str*) – Default linestyle used in *matplotlib.pyplot.plot*. The difference is that now it accepts `linestyle='exposure'`, where the plot will be a dashed line corresponding to each exposure. The blank space between the lines can be interpreted as 'dead time'.
- **time_direction** (*bool*) – If enabled, it draws the direction of time flow on the chord line.
- ****kwargs** – Any other kwarg will be parsed directly by *maplotlip.pyplot.plot*. The only difference is that the default linewidth `lw=2`.

status()

Returns if the chord is positive or negative

6.6.3 The Chordlist Class

class sora.occultation.chordlist.**ChordList**(*, *star*, *body*, *time*)

Defines a collection of Chord objects associated to an Occultation.

This object is not supposed to be defined by the user. It will be automatically defined in Occultation.

star

The Star occulted.

Type
sora.Star

body

The occulting Body.

Type
sora.Body

time

The occultation time.

Type
astropy.time.Time

add_chord(*, *name=None*, *chord=None*, *observer=None*, *lightcurve=None*)

Add a chord to the occultation chord list

Parameters

- **name** (*str*) – The name of the Chord. It must be unique within the list of chords. If not given, it will use the name in chord if chord is directly given or the name in the observer object. If the name in Chord already exists in the list, the name parameter can be given to update the chord name. The *name* parameter is required if given together with observer and lightcurve. It can not be an empty string.
- **sora.occultation.Chord** (*chord*) – A chord instance defined by the user.
- **observer** (*sora.Observer*) – An Observer instance defined by the user. It must be given together with a lightcurve.
- **lightcurve** (*sora.LightCurve*) – A lightcurve defined by the user. It must be given together with observer.

Examples

Ways to add a chord:

```
>>> obj.add_chord(chord) # where the name in chord will be used.
```

```
>>> obj.add_chord(name, chord) # where the name in chord will be replaced by
↪"name".
```

```
>>> obj.add_chord(observer, lightcurve) # where the name in observer will be␣
↪used.
```

```
>>> obj.add_chord(name, observer, lightcurve)
```

`disable`(*, *chord=None*, *time=None*)

Disable a contact point of the curve to be used in the fit.

Parameters

- **chord** (*str*) – Name of the chord to disable. If `chord=None`, it applies to all chords.
- **time** (None, *str*) – If `time=None`, it will disable all contact points. If 'immersion' or 'emersion', it will disable respective contact point.

`enable`(*, *chord=None*, *time=None*)

Enable a contact point of the curve to be used in the fit.

Parameters

- **chord** (*str*) – Name of the chord to enable. If `chord=None`, it applies to all chords.
- **time** (Non`, *str*) – If `time=None`, it will enable all contact points. If 'immersion' or 'emersion', it will enable respective contact point.

`get_impact_param`(*chords='all_chords'*, *center_f=0*, *center_g=0*, *verbose=True*)

Get the impact parameter, minimal distance between the chord and the centre position.

This Chord object must be associated to an Occultation to work, since it needs the position of the star and an ephemeris.

Parameters

- **chords** (*int*, *str*, default='all_chords') – Index or names of the chords to be considered.
- **center_f** (*int*, *float*, default=0) – The coordinate in f of the ellipse center.
- **center_g** (*int*, *float*, default=0) – The coordinate in g of the ellipse center.
- **verbose** (*bool*) – If True, prints the obtained values.

Returns

impact, sense, chord_name – The impact parameter (in km), the direction of the chord relative the ellipse center, North (N), South (S), East (E) and West (W), and the name of the chord

Return type

list

`get_limb_points`(*only_able=True*)

Computes the projected points and errors on the tangent plane for every chord

Parameters

only_able (*bool*) – Get only the contact points that are able to be used in the fit.

Returns

- **fg** (*numpy.array*) – The projected points of occultation instants. Each line is a point on the projection with x and y respectively.
- **error** (*numpy.array*) – Error vector of the projected occultation instants. Each line is the error of a point in x and y, respectively.

`get_theoretical_times`(*equatorial_radius*, *chords='all_chords'*, *center_f=0*, *center_g=0*, *oblateness=0*, *position_angle=0*, *sigma=0*, *step=1*, *verbose=True*)

Get the theoretical times and chords sizes for a given ellipse.

This Chord object must be associated to an Occultation to work, since it needs the position of the star and an ephemeris.

Parameters

- **chords** (*int*, *str*, default='all_chords') – Index or names of the chords to be considered.
- **equatorial_radius** (*int*, *float*) – The Equatorial radius (semi-major axis) of the ellipse.
- **center_f** (*int*, *float*, default=0) – The coordinate in f of the ellipse center.
- **center_g** (*int*, *float*, default=0) – The coordinate in g of the ellipse center.
- **oblateness** (*int*, *float*, default=0) – The oblateness of the ellipse.
- **position_angle** (*int*, *float*, default=0) – The pole position angle of the ellipse in degrees. Zero is in the North direction ('g-positive'). Positive clockwise.
- **sigma** (*int*, *float*) – Uncertainty of the expected ellipse, in km.
- **step** (*int*, *float*) – Time resolution of the chord, in seconds.
- **verbose** (*bool*) – If True, prints the obtained values.

Returns

theory_immersion_time, theory_emersion_time, theory_chord_size, chord_name – The expected immersion time for the given ellipse, the expected emersion time for the given ellipse, the expected chord size for the given ellipse, and the name of the chord.

Return type

list

plot_chords(*, *segment='standard'*, *ignore_chords=None*, *only_able=False*, *ax=None*, *linestyle='-'*, ***kwargs*)

Plots the on-sky path of this chord.

Parameters

- **segment** (*str*) – The segment to plot the chord. The available options are:

  `'positive'` to get the path between the immersion and emersion times if the chord is positive.

  `'negative'` to get the path between the start and end of observation if the chord is negative.

  `'standard'` to get the 'positive' path if the chord is positive or 'negative' if the chord is negative.

  `'full'` to get the path between the start and end of observation independent if the chord is positive or negative.

  `'outer'` to get the path outside the 'positive' path, for instance between the start and immersion times and between the emersion and end times.

  `'error'` to get the path corresponding to the error bars.
- **ignore_chords** (*str*, *list*) – Name of chord or list of names to ignore in the plot.
- **only_able** (*bool*) – Plot only the chords or contact points that are able to be used in the fit. If `segment='error'` it will show only the contact points able. If segment is any other, the path will be plotted only if both immersion and emersion are able, or it is a negative chord.
- **ax** (*matplotlib.pyplot.Axes*) – The axes where to make the plot. If None, it will use the default axes.
- **linestyle** (*str*) – Default linestyle used in *matplotlib.pyplot.plot*. The difference is that now it accept `linestyle='exposure'`, where the plot will be a dashed line corresponding to each exposure. The blank space between the lines can be interpreted as 'dead time'.

- ****kwargs** – Any other kwarg will be parsed directly by *maplotlip.pyplot.plot*. The only difference is that the default linewidth `lw=2`.

remove_chord(*, *name*)

Remove a chord from the chord list and disassociate it from the Occultation.

Parameters

name (*str*) – The name of the chord.

summary()

Prints a table with the summary of the chords.

6.6.4 Fit Ellipse

sora.occultation.**fit_ellipse**(**args*, *equatorial_radius*, *dequatorial_radius=0*, *center_f=0*, *dcenter_f=0*, *center_g=0*, *dcenter_g=0*, *oblateness=0*, *doblateness=0*, *position_angle=0*, *dposition_angle=0*, *loop=10000*, *number_chi=10000*, *dchi_min=None*, *verbose=True*, *ellipse_error=0*, *sigma_result=1*, *method='least_squares'*, *threads=1*)

Fits an ellipse to given occultation using given parameters.

Parameters

- **center_f** (*int*, *float*, default=0) – The coordinate in f of the ellipse center.
- **center_g** (*int*, *float*, default=0) – The coordinate in g of the ellipse center.
- **equatorial_radius** (*int*, *float*) – The Equatorial radius (semi-major axis) of the ellipse.
- **oblateness** (*int*, *float*, default=0) – The oblateness of the ellipse.
- **position_angle** (*int*, *float*, default=0) – The pole position angle of the ellipse in degrees. Zero is in the North direction ('g-positive'). Positive clockwise.
- **dcenter_f** (*int*, *float*) – Interval for coordinate f of the ellipse center.
- **dcenter_g** (*int*, *float*) – Interval for coordinate g of the ellipse center.
- **int** (*dequatorial_radius*) – Interval for the Equatorial radius (semi-major axis) of the ellipse.
- **float** – Interval for the Equatorial radius (semi-major axis) of the ellipse.
- **doblateness** (*int*, *float*) – Interval for the oblateness of the ellipse
- **dposition_angle** (*int*, *float*) – Interval for the pole position angle of the ellipse in degrees.
- **loop** (*int*, default=10000000) – The number of ellipses to attempt fitting.
- **dchi_min** (*int*, *float*) – If given, it will only save ellipsis which chi square are smaller than chi_min + dchi_min. By default *None* when used with *method='chisqr'*, and *3* for other methods.
- **number_chi** (*int*, default=10000) – In the *chisqr* method if *dchi_min* is given, the procedure is repeated until *number_chi* is reached. In other methods it is the number of values (simulations) that should lie within the provided *sigma_result*.
- **verbose** (*bool*, default=True) – If True, it prints information while fitting.
- **ellipse_error** (*int*, *float*) – Model uncertainty to be considered in the fit, in km.
- **sigma_result** (*int*, *float*) – Sigma value to be considered as result.

- **method** (*str*, *default='least_squares'*) – Method used to perform the fit. Available methods are: *chisqr* : monte carlo computation method used in versions of SORA <= 0.2.1. *fastchi* : monte carlo computation method, allows multithreading . *least_squares* or *ls*: best fit done used levenberg marquardt convergence algorithm. *differential_evolution* or *de*: best fit done using genetic algorithms. All methods return a Chisquare object.
- **threads** (*int*) – Number of threads/workers used to perform parallel computations of the chi square object. It works with all methods except *chisqr*, by default 1.

Returns
chisquare – A ChiSquare object with all parameters.

Return type
sora.ChiSquare

Important: Each occultation is added as the first argument(s) directly.

Mandatory input parameters: 'center_f', 'center_g', 'equatorial_radius', 'oblateness', and 'position_angle'.

Parameters fitting interval: 'dcenter_f', 'dcenter_g', 'dequatorial_radius', 'doblateness', and 'dposition_angle'. Default values are set to zero. Search done between (value - dvalue) and (value + dvalue).

Examples

To fit the ellipse to the chords of occ1 Occultation object:

```
>>> fit_ellipse(occ1, **kwargs)
```

To fit the ellipse to the chords of occ1 and occ2 Occultation objects together:

```
>>> fit_ellipse(occ1, occ2, **kwargs)
```

6.6.5 Complementary functions

sora.occultation.utils.**add_arrow**(*line*, *position=None*, *direction='right'*, *size=15*, *color=None*)

Add an arrow to a chord.

Parameters

- **line** (*Line2D object*) – Line2D object
- **position** (*float*, *int*) – x-position of the arrow. If None, mean of xdata is taken.
- **direction** (*string*, default='right') – 'left' or 'right'
- **size** (*float*, *int*, default=15) – Size of the arrow in fontsize points.
- **color** (*string*, default=None) – If None, line color is taken.

See also:

https
//stackoverflow.com/a/34018322/3137585

sora.occultation.utils.**filter_negative_chord**(*chord*, *chisquare*, *step=1*, *sigma=0*)

Get points for the ellipse with the given input parameters.

Parameters

- **chord** (*sora.observer.Chord*) – Chord object, must be associated to an Occultation to work.
- **chisquare** (*sora.extra.ChiSquare*) – Resulted ChiSquare object of fit_ellipse.
- **sigma** (*int*, *float*) – Uncertainty of the expected ellipse, in km.
- **step** (*int*, *float*, *str*) – If a number, it corresponds to the step, in seconds, for each point of the chord path. The step can also be equal to `'exposure'`. In this case, the chord path will consider the lightcurve individual times and exptime.

sora.occultation.utils.**positionv**(*star*, *ephem*, *observer*, *time*)

Calculates the position and velocity of the occultation shadow relative to the observer.

Parameters

- **star** (*sora.Star*) – The coordinate of the star in the same reference frame as the ephemeris. It must be a Star object.
- **ephem** (*sora.Ephem*) – The object ephemeris. It must be an Ephemeris object.
- **observer** (*sora.Observer*) – The Observer information. It must be an Observer object.
- **time** (*astropy.time.Time*) – Reference instant to calculate position and velocity. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

Returns

f, g, vf, vg – The orthographic projection of the shadow relative to the observer. `'f'` is in the x-axis (East-West direction; East positive). `'g'` is in the y-axis (North-South direction; North positive).

Return type

list

6.7 sora.prediction

6.7.1 Prediction Functions

sora.prediction.core.**occ_params**(*star*, *ephem*, *time*, *n_recursions=5*, *max_tdiff=None*, *reference_center='geocenter'*)

Calculates the parameters of the occultation, as instant, CA, PA.

Parameters

- **star** (*sora.Star*) – The coordinate of the star in the same reference frame as the ephemeris. It must be a Star object.
- **ephem** (*sora.Ephem**) – Object ephemeris. It must be an Ephemeris object.
- **time** (*astropy.time.Time*) – Time close to occultation epoch to calculate occultation parameters.
- **n_recursions** (*int*, default=5) – The number of attempts to try obtain prediction parameters in case the event is outside the previous range of time.
- **max_tdiff** (*int*, default=None) – Maximum difference from given time it will attempt to identify the occultation, in minutes. If given, 'n_recursions' is ignored.

- **reference_center** (*str*, *sora.Observer*, *sora.Spacecraft*) – A SORA observer object or a string 'geocenter'. The occultation parameters will be calculated in respect to this reference as center of projection.

Returns

Oredered list –

- Instant of CA (Time): Instant of Closest Approach.
- CA (arcsec): Distance of Closest Approach.
- PA (deg): Position Angle at Closest Approach.
- vel (km/s): Velocity of the occultation.
- dist (AU): the object geocentric distance.

Return type
list

sora.prediction.core.**prediction**(*time_beg*, *time_end*, *body=None*, *ephem=None*, *mag_lim=None*, *catalogue='gaiadr3'*, *step=60*, *divs=1*, *sigma=1*, *radius=None*, *verbose=True*, *reference_center='geocenter'*)

Predicts stellar occultations.

Parameters

- **time_beg** (*str*, *astropy.time.Time*, required) – Initial time for prediction.
- **time_end** (*str*, *astropy.time.Time*, required) – Final time for prediction.
- **body** (*sora.Body*, *str*, default=None) – Object that will occult the stars. It must be a Body object or its name to search in the Small Body Database.
- **ephem** (*sora.Ephem*, default=None) – object ephemeris. It must be an Ephemeris object. If using a EphemHorizons object, please use 'divs' to make division at most a month, or a timeout error may be raised by the Horizon query.
- **mag_lim** (*int*, *float*, *dict*, default=None) – Faintest magnitude allowed in the search. If the catalogue has more than one band defined in the catalogue object, the magnitude limit can be done for a specific band or a set of band. Ex: `mag_lim={'V': 15}`, which will only download stars with V<=15 or `mag_lim={'V': 15, 'B': 14}` which will download stars with V<=15 AND B<=14.
- **catalogue** (*str*, *VizierCatalogue*) – The catalogue to download data. It can be `'gaiadr2'`, `'gaiaedr3'`, `'gaiadr3'`, or a VizierCatalogue object. default='gaiadr3'
- **step** (*int*, *float*, default=60) – Step, in seconds, of ephem times for search
- **divs** (*int*, default=1) – Number of regions the ephemeris will be split for better search of occultations.
- **sigma** (*int*, *float*, default=1) – Ephemeris error sigma for search off-Earth.
- **radius** (*int*, *float*, default=None) – The radius of the body. It is important if not defined in body or ephem.
- **verbose** (*bool*, default=True) – To show what is being done at the moment.
- **reference_center** (*str*, *sora.Observer*, *sora.Spacecraft*) – A SORA observer object or a string 'geocenter'. The occultation parameters will be calculated in respect to this reference as center of projection. If a Spacecraft is used, please use smaller step since the search will be based on the target size and ephemeris error only.

Important: When instantiating with "body" and "ephem", the user may call the function in 3 ways:

1 - With "body" and "ephem".

2 - With only "body". In this case, the "body" parameter must be a Body object and have an ephemeris associated (see Body documentation).

3 - With only "ephem". In this case, the "ephem" parameter must be one of the Ephem Classes and have a name (see Ephem documentation) to search for the body in the Small Body Database.

Returns
PredictionTable with the occultation params for each event.

Return type
sora.prediction.PredictionTable

6.7.2 Plot Occultation Map

`sora.prediction.occmap.`**`plot_occ_map`**(*name*, *radius*, *coord*, *time*, *ca*, *pa*, *vel*, *dist*, *mag=0*, *longi=0*, ***kwargs*)

Plots the map of the occultation.

Parameters

- **`name`** (*str*) – Name of the object.
- **`radius`** (*int*, *float*) – Radius of the object, in km.
- **`coord`** (*str*, *astropy.coordinates.SkyCoord*) – Coordinates of the star ("`hh mm ss.sss dd mm ss.sss`" or "`hh.hhhhhhhh dd.dddddddd`").
- **`time`** (*str*, *astropy.time.Time*) – Instant of Closest Approach (iso or isot format).
- **`ca`** (*int*, *float*) – Closest Approach Distance, in arcsec.
- **`pa`** (*int*, *float*) – Position Angle at C/A, in degrees.
- **`vel`** (*int*, *float*) – Velocity of the event, in km/s.
- **`dist`** (*int*, *float*) – Object distance at C/A, in AU.
- **`mag`** (*int*, *float*, default=0) – Mag* = Normalized magnitude to vel=20km/s.
- **`longi`** (*int*, *float*, default=0) – East longitude of sub-planet point, deg, positive towards East.
- **`nameimg`** (*str*) – Change the name of the imaged saved.
- **`path`** (*str*) – Path to a directory where to save map.
- **`resolution`** (*int*, default=2) – Cartopy feature resolution.
  - `1` means a resolution of "10m";
  - `2` a resolution of "50m";
  - `3` a resolution of "100m".
- **`states`** (*bool*) – If True, plots the states borders of the countries. The states of some countries will only be shown depending on the resolution.
- **`zoom`** (*int*, *float*) – Zooms in or out of the map.

- **centermap_geo** (*list*, default=None) – Center the map given coordinates in longitude and latitude. It must be a list with two numbers.
- **centermap_delta** (*list*, default=None) – Displace the center of the map given displacement in X and Y, in km. It must be a list with two numbers.
- **centerproj** (*list*) – Rotates the Earth to show occultation with the center projected at a given longitude and latitude. It must be a list with two numbers.
- **labels** (*bool*, default=True) – Plots text above and below the map with the occultation parameters.
- **meridians** (*int*, default=30) – Plots lines representing the meridians for given interval, in degrees.
- **parallels** (*int*, default=30) – Plots lines representing the parallels for given interval, in degrees.
- **sites** (*dict*) – Plots site positions in map. It must be a python dictionary where the key is the *name* of the site, and the value is a list with *longitude*, *latitude*, *delta_x*, *delta_y* and *color*. *delta_x* and *delta_y* are displacement, in km, from the point position of the site in the map and the *name*. *color* is the color of the point.
- **site_name** (*bool*) – If True, it prints the name of the sites given, else it plots only the points.
- **site_box_alpha** (*int*, *float*, default=0) – Sets the transparency of a box surrounding each station name. 0 equals to transparent, and 1 equals to opaque.
- **countries** (*dict*) – Plots the names of countries. It must be a python dictionary where the key is the name of the country and the value is a list with longitude and latitude of the lower left part of the text.
- **offset** (*list*) – Applies an offset to the ephemeris, calculating new CA and instant of CA. It is a pair of *delta_RA*cosDEC* and *delta_DEC*.
- **mapstyle** (*int*, default=1) – Define the color style of the map. `'1'` is the default black and white scale. `'2'` is a colored map.
- **error** (*int*, *float*) – Ephemeris error in mas. It plots a dashed line representing radius + error.
- **ercolor** (*str*) – Changes the color of the lines of the error bar.
- **ring** (*int*, *float*) – Plots a dashed line representing the location of a ring. It is given in km, from the center.
- **rncolor** (*str*) – Changes the color of ring lines.
- **atm** (*int*, *float*) – Plots a dashed line representing the location of an atmosphere. It is given in km, from the center.
- **atcolor** (*str*) – Changes the color of atm lines.
- **chord_delta** (*list*) – List with distances from center to plot chords.
- **chord_geo** (*2d-list*) – List with pairs of coordinates to plot chords.
- **chcolor** (*str*, default='grey') – Color of the line of the chords.
- **heights** (*list*) – It plots a circular dashed line showing the locations where the observer would observe the occultation at a given height above the horizons. This must be a list.
- **hcolor** (*str*) – Changes the color of the height lines.
- **mapsize** (*list*, default= [46.0, 38.0]) – The size of figure, in cm. It must be a list with two values.

- **cpoints** (*int*, *float*, default=60) – Interval for the small points marking the center of shadow, in seconds.
- **ptcolor** (*str*) – Change the color of the center points.
- **alpha** (*float*, default=0.2) – The transparency of the night shade, where 0.0 is full transparency and 1.0 is full black.
- **fmt** (*str*, default:'png') – The format to save the image. It is parsed directly by *matplotlib.pyplot*.
- **dpi** (*int*, default=100) – Resolution in "dots per inch". It defines the quality of the image.
- **lncolor** (*str*) – Changes the color of the line that represents the limits of the shadow over Earth.
- **outcolor** (*str*) – Changes the color of the lines that represents the limits of the shadow outside Earth.
- **scale** (*int*, *float*) – Arbitrary scale for the size of the name of the site.
- **cscale** (*int*, *float*) – Arbitrary scale for the name of the country.
- **sscale** (*int*, *float*) – Arbitrary scale for the size of point of the site.
- **pscale** (*int*, *float*) – Arbitrary scale for the size of the points that represent the center of the shadow.
- **arrow** (*bool*) – If True, it plots the arrow with the occultation direction.

Important: Required parameters to plot an occultation map: 'name', 'radius', 'coord', 'time', 'ca', 'pa', 'vel', and 'dist'.

Note: The parameters 'mag' and 'longi' are optional and only printed in label. All other remaining parameters can be used to further customize the Map configuration.

When producing the map, only one of 'centermap_geo' or 'centermap_delta' options can be used at a time.

6.7.3 PredictionTable Cbject

class sora.prediction.**PredictionTable**(**args*, ***kwargs*)

An Astropy Table object modified for Prediction purposes.

Row

alias of `PredictRow`

classmethod from_praia(*filename*, *name*, ***kwargs*)

Creates a PredictionTable Table reading from a PRAIA table.

Parameters

- **filename** (*str*) – Path to the PRAIA table file.
- **name** (*str*) – Name of the Object of the prediction.
- **radius** (*int*, *float*, optional) – Object radius, in km. If not given it's searched in online database. When not found online, the default is set to zero.

Returns
A PredictionTable object.

Return type
sora.prediction.PredictionTable

`keep_from_selected_images`(*path='.'*)

Keeps predictions which images were not deleted in given path. This function uses the name of the images to identify predictions. The name must be the automatic one generated by plot_occ_map(). The format of the image is not relevant.

Parameters
path (*str*) – Path where images are located.

`plot_occ_map`(***kwargs*)

Parameters

- **radius** (*int*, *float*) – The radius of the shadow. If not given it uses saved value.
- **nameimg** (*str*) – Change the name of the image saved.
- **path** (*str*) – Path to a directory where to save map.
- **resolution** (*int*, default=2) –
- **resolution.** (*Cartopy feature*) –
- **"10m";** (*- 1 means a resolution of*) –
- **"50m";** (*- 2 a resolution of*) –
- **"100m".** (*- 3 a resolution of*) –
- **states** (*bool*) – If True, plots the states borders of the countries. The states of some countries will only be shown depending on the resolution.
- **zoom** (*int*, *float*) – Zooms in or out of the map.
- **centermap_geo** (*list*, default=None) – Center the map given coordinates in longitude and latitude. It must be a list with two numbers.
- **centermap_delta** (*list*, default=None) – Displace the center of the map given displacement in X and Y, in km. It must be a list with two numbers.
- **centerproj** (*list*) – Rotates the Earth to show occultation with the center projected at a given longitude and latitude. It must be a list with two numbers.
- **labels** (*bool*, default=True) – Plots text above and below the map with the occultation parameters.
- **meridians** (*int*, default=30) – Plots lines representing the meridians for given interval, in degrees.
- **parallels** (*int*, default=30) – Plots lines representing the parallels for given interval, in degrees.
- **sites** (*dict*) – Plots site positions in map. It must be a python dictionary where the key is the *name* of the site, and the value is a list with *longitude*, *latitude*, *delta_x*, *delta_y* and *color*. *delta_x* and *delta_y* are displacement, in km, from the point position of the site in the map and the *name*. *color* is the color of the point.
- **site_name** (*bool*) – If True, it prints the name of the sites given, else it plots only the points.

- **countries** (*dict*) – Plots the names of countries. It must be a python dictionary where the key is the name of the country and the value is a list with longitude and latitude of the lower left part of the text.
- **offset** (*list*) – Applies an offset to the ephemeris, calculating new CA and instant of CA. It is a pair of *delta_RA*cosDEC* and *delta_DEC*.
- **mapstyle** (*int*, default=1) – Define the color style of the map. `'1'` is the default black and white scale. `'2'` is a colored map.
- **error** (*int*, *float*) – Ephemeris error in mas. It plots a dashed line representing radius + error.
- **ercolor** (*str*) – Changes the color of the lines of the error bar.
- **ring** (*int*, *float*) – Plots a dashed line representing the location of a ring. It is given in km, from the center.
- **rncolor** (*str*) – Changes the color of ring lines.
- **atm** (*int*, *float*) – Plots a dashed line representing the location of an atmosphere. It is given in km, from the center.
- **atcolor** (*str*) – Changes the color of atm lines.
- **chord_delta** (*list*) – List with distances from center to plot chords.
- **chord_geo** (*2d-list*) – List with pairs of coordinates to plot chords.
- **chcolor** (*str*, default='grey') – Color of the line of the chords.
- **heights** (*list*) – It plots a circular dashed line showing the locations where the observer would observe the occultation at a given height above the horizons. This must be a list.
- **hcolor** (*str*) – Changes the color of the height lines.
- **mapsize** (*list*, default= [46.0, 38.0]) – The size of figure, in cm. It must be a list with two values.
- **cpoints** (*int*, *float*, default=60) – Interval for the small points marking the center of shadow, in seconds.
- **ptcolor** (*str*) – Change the color of the center points.
- **alpha** (*float*, default=0.2) – The transparency of the night shade, where 0.0 is full transparency and 1.0 is full black.
- **fmt** (*str*, default:'png') – The format to save the image. It is parsed directly by *matplotlib.pyplot*.
- **dpi** (*int*, default=100) – Resolution in "dots per inch". It defines the quality of the image.
- **lncolor** (*str*) – Changes the color of the line that represents the limits of the shadow over Earth.
- **outcolor** (*str*) – Changes the color of the lines that represents the limits of the shadow outside Earth.
- **scale** (*int*, *float*) – Arbitrary scale for the size of the name of the site.
- **cscale** (*int*, *float*) – Arbitrary scale for the name of the country.
- **sscale** (*int*, *float*) – Arbitrary scale for the size of point of the site.
- **pscale** (*int*, *float*) – Arbitrary scale for the size of the points that represent the center of the shadow.

- **arrow** (*bool*) – If True, it plots the arrow with the occultation direction.

Note: Only one of centermap_geo and centermap_delta can be given.

remove_occ(*date*)

Removes stellar occultations from table.

Parameters

date (*str*, *list*) – Date or list of dates of the occultation to be removed. The dates mut be as shown in the 'Epoch' column. If the date is not complete, the function will select all occultations that matches the given string. For instance, `date='2020-06'` will remove all occultations from the month of June 2020.

to_ow(*ow_des*, *mode='append'*)

Writes PredictionTable to OccultWatcher feeder update file format. Tables will be saved in two files: "tableOccult_update.txt" and "LOG.dat"

Parameters

- **ow_des** (*str*) – Occult Watcher designation for the object.
- **mode** (*str*, default='append') – Use `'append'` to append table to already existing file and `'restart'` to overwrite existing file.

to_praia(*filename*)

Writes PredictionTable to PRAIA format.

Parameters

filename (*str*) – Name of the file to save table.

6.8 sora.star

6.8.1 The Star Class

class sora.**Star**(*catalogue='gaiadr3'*, ***kwargs*)

Defines a star.

Parameters

- **catalogue** (*str*, *VizierCatalogue*) – The catalogue to download data. It can be `'gaiadr2'`, `'gaiaedr3'`, `'gaiadr3'`, or a VizierCatalogue object.. default='gaiadr3'
- **code** (*str*) – Gaia Source code for searching in VizieR.
- **coord** (*str*, *astropy.coordinates.SkyCoord*) – If code is not given, coord nust have the coordinates RA and DEC of the star to search in VizieR: `'hh mm ss.ss +dd mm ss.ss'`.
- **ra** (*int*, *float*) – Right Ascension, in deg.
- **dec** (*int*, *float*) – Declination, in deg.
- **parallax** (*int*, *float*. default=0) – Parallax, in mas.
- **pmra** (*int*, *float*, default=0) – Proper Motion in RA*, in mas/year.
- **pmdec** (*int*, *float*, default=0) – Proper Motion in DEC, in mas/year.
- **rad_vel** (*int*, *float*, default=0) – Radial Velocity, in km/s.

- **epoch** (*str*, *astropy.time.Time*, default='J2000') – Epoch of the coordinates.
- **nomad** (*bool*) – If True, it tries to download the magnitudes from NOMAD catalogue.
- **bjones** (*bool*, default=True) – If True, it uses de star distance from Bailer-Jones et al. (2018).
- **cgaudin** (*bool*, default=True) – If True, it uses de proper motion correction from Cantat-Gaudin & Brandt (2021). this option is only available for Gaia-EDR3.
- **verbose** (*bool*, default=True) – If True, it prints the downloaded information
- **local** (*bool*, default=False) – If True, it uses the given coordinate in 'coord' as final coordinate.

Note: The user can give either 'coord' or 'ra' and 'dec', but not both.

To download the coordinates from Gaia, "local" must be set as False and the ("code") or ("coord") or ("ra" and "dec") must be given.

All values downloaded from Gaia will replace the ones given by the user.

`add_offset`(*da_cosdec*, *ddec*)

Adds an offset to the star position.

Parameters

- **da_cosdec** (*int*, *float*) – Offset in Delta_alpha_cos_delta, in mas.
- **ddec** (*int*, *float*) – Offset in Delta_delta, in mas.

`apparent_diameter`(*distance*, *mode='auto'*, *band='V'*, *star_type='sg'*, *verbose=True*)

Calculates the apparent diameter of the star at a given distance.

Parameters

- **distance** (*int*, *float*) – Object geocentric distance, in AU.
- **mode** (*str*, default='auto') – The mode to calculate the apparent diameter.
  - `'user'`: calculates using user given diameter.
  - `'gaia'`: calculates using diameter obtained from Gaia.
  - `'kervella'`: calculates using Kervella equations.
  - `'van_belle'`: calculates using van Belle equations.
  - `'auto'`: tries all the above methods until it is able to calculate diameter.

  The order of try is the same as shown above (user, Gaia, Kervella, Van Belle).
- **band** (*str*) – The band filter to calculate the diameter. If mode is *kervella* or *van_belle*, the filter must be given, `'B'` or `'V'`. If mode *auto*, `'V'` is selected.
- **star_type** (*str*) – Type of star to calculate the diameter. If mode is *van_belle*, the star type must be given. If mode is *auto*, `star_type='sg'`.

  Accepted types:
  - `'sg'` for 'Super Giant'.
  - `'ms'` for 'Main Sequence'.
  - `'vs'` for 'Variable Star'.
- **verbose** (*bool*) – If True, it prints the mode used by *auto*.

barycentric(*time*)

Calculates the position of the star using proper motion.

Parameters
time (*str*, *astropy.time.Time*) – Reference time to apply proper motion. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

error_at(*time*)

Estimates the star position error at a given time.

Parameters
time (*str*, *astropy.time.Time*) – Reference time to project star error. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

Returns
errors – In RA* and DEC.

Return type
list

geocentric(*time*)

Calculates the position of the star, propagating the position using parallax and proper motion

Parameters
time (*str*, *astropy.time.Time*) – Reference time to apply proper motion and calculate parallax. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

get_position(*time*, *observer='geocenter'*)

Calculates the position of the star for given observer, propagating the position using parallax and proper motion

Parameters

- **time** (*float*, *astropy.time.Time*) – Reference time to apply proper motion and calculate parallax. It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.
- **observer** (*str*, *sora.observer.Observer*, *sora.observer.Spacecraft*) – Observer of the star t calculate position. It can be 'geocenter' for a geocentric coordinate, 'barycenter' for a barycenter coordinate, or a sora observer object.

Returns
coord – Astropy SkyCoord object with the star coordinates at the given time.

Return type
astropy.coordinates.SkyCoord

kervella()

Determines the diameter of a star in mas using equations from Kervella et. al (2004).

See: A&A Vol. 426, No. 1:.

set_diameter(*diameter*)

Sets an user diameter for the star, in mas.

Parameters
diameter (*int*, *float*) – Sets the user diameter of the star, in mas.

set_magnitude(***kwargs*)

Sets the magnitudes of a star.

Parameters
band=value (*str*) – The star magnitude for given band. The band name can be any string the user wants.

Examples

To set the stars magnitude in the band G:

```
>>> set_magnitude(G=10)
```

To set the star's magnitude in the band K:

```
>>> set_magnitude(K=15)
```

To set the star's magnitude in a customized band:

```
>>> set_magnitude(newband=6)
```

to_log(*namefile*)

Saves the star log to a file.

Parameters
namefile (*str*) – Filename to save the log.

van_belle()

Determines the diameter of a star in mas using equations from van Belle (1999).

See: Publi. Astron. Soc. Pacific 111, 1515-1523:.

6.8.2 The Star Catalogue

class sora.star.catalog.**VizierCatalogue**(***kwargs*)

VizierCatalogue defines the parameters necessary to download all the information of stars from a catalogue on the Vizier webservice.

Parameters

- **name** (*str*) – The name of the catalogue which will be referred in other processes
- **cat_path** (*str*) – The path of the catalogue in the Vizier website. For instance, for GaiaEDR3, `cat_path='I/350/gaiaedr3'`
- **code** (*str*) – The keyword referring to a unique code within the catalogue
- **ra** (*str*) – The keyword referring to the Right Ascension within the catalogue
- **dec** (*str*) – The keyword referring to the Declination within the catalogue
- **epoch** (*str*, *astropy.time.Time*) – The epoch of the catalogue. If it is defined in the catalogue, just pass the keyword within the catalogue. If the epoch is not present in the catalogue table, we must pass a Time object directly, for example `epoch=Time('J2000')`, which defines the catalogue coordinates in J2000 TDB.
- **pmra** (*str*, optional) – The keyword referring to the Proper Motion in ra*cosdec within the catalogue. If not available, set it to None.
- **pmdec** (*str*, optional) – The keyword referring to the Proper Motion in dec within the catalogue. If not available, set it to None.

- **parallax** (*str*, optional) – The keyword referring to the Parallax within the catalogue. If not available, set it to None.
- **rad_vel** (*str*, optional) – The keyword referring to the Radial Velocity within the catalogue
- **bands** (*dict* [*str*, *str*], optional) – A dictionary where the key is band name and the value is the keyword referring to the band within the catalogue. For instance, in Gaia: `bands={'G': 'Gmag'}`. If not available, set it to None.
- **errors** (*list* [*str*], optional) – A list with the 6 keywords that refer to the uncertainty parameters within the catalogue in the order: [ra, dec, pmra, pmdec, parallax, rad_vel]. If some parameters are not available, please pass each one as None. Ex: `errors=['eRA', 'eDEC', None, None, None, None]`, or `errors=None` if none of the errors is available.

Examples

To define the Gaia-EDR3 catalogue with VizierCatalogue, we must define a object like:

```
>>> catalogue = VizierCatalogue(name='GaiaEDR3', cat_path='I/350/gaiaedr3', code=
↪'Source', ra='RA_ICRS', dec='DE_ICRS',
>>>                             pmra='pmRA', pmdec='pmDE', epoch='Epoch', parallax=
↪'Plx', rad_vel='RVDR2', band={'G': 'Gmag'},
>>>                             errors=['e_RA_ICRS', 'e_DE_ICRS', 'e_pmRA', 'e_pmDE
↪', 'e_Plx', 'e_RVDR2'])
```

parse_catalogue(*table*)

Properly interprets the table

Parameters
table (*astropy.table.Table*) – The table with the parameters read from the catalogue server

Returns
cat_info – Dictionary with the list of parameters

Return type
dict

search_region(*coord*, *radius=None*, *width=None*, *height=None*, *columns=None*, *row_limit=10000000*, *timeout=600*, ***kwargs*)

Parameters

- **coord** (*str*, *astropy.coordinates.SkyCoord*) – The target around which to search. It may be specified as a string in which case it is resolved using online services or as the appropriate astropy SkyCoord object. ICRS coordinates may also be entered as a string.
- **radius** (*number*) – The radius of the circular region to query.
- **width** (*number*) – The width in of the square region to query.
- **height** (*number*) – When set in addition to `width`, the queried region becomes rectangular, with the specified `width` and `height`.
- **columns** (*list*) – List of strings with the keyword to fetch the catalog. If `columns=None` it will download all the columns. If `columns="simple"` it will download only the columns for the code, epoch and astrometric parameters.
- **row_limit** (*int*) – Maximum number of rows that will be fetched from the result (set to -1 for unlimited). Default: `row_limit=10_000_000`

- **timeout** (*number*) – timeout for connecting to server in seconds. Default: `timeout=600`
- ****kwargs** – Any other keyword argument will be passed to astroquery.vizier.Vizier

Returns
catalogue – An astropy Table with all the information about the star

Return type
astropy.table.Table

`search_star`(*code=None*, *coord=None*, *radius=None*)

Looks for a specific star in the catalogue

Parameters

- **code** (*str*) – The unique id of the star
- **coord** (*str*, *astropy.coordinates.SkyCoord*) – The target coordinate which to search. It may be specified as a string in which case it is resolved using online services or as the appropriate astropy SkyCoord object. ICRS coordinates may also be entered as a string.
- **radius** (*number*) – The radius of the circular region to query.

Returns
catalogue – An astropy Table with all the information about the star

Return type
astropy.table.Table

Raises
`ValueError` – Raised when `code` or (`coord`, `radius`) are not provided.

Notes

This function must be called in one of the following ways:

- Using `code` if the unique id of the star is known
- Using `coord` **and** `radius` if the catalogue position is known.

If both alternatives are provided, only the first is used.

6.8.3 Complementary functions

`sora.star.utils.`**kervella**(*magB=None*, *magV=None*, *magK=None*)

Determines the diameter of a star in mas using equations from Kervella et. al (2004).

See: A&A Vol. 426, No. 1:.

Parameters

- **magB** (*float*, default=None) – The magnitude B of the star.
- **magV** (*float*, default=None) – The magnitude V of the star.
- **magK** (*float*, default=None) – The magnitudes K of the star.

Note: If any of those values is 'None', 'nan' or higher than 49, it is not considered.

sora.star.utils.**van_belle**(*magB=None*, *magV=None*, *magK=None*)

Determines the diameter of a star in mas using equations from van Belle (1999).

See: Publi. Astron. Soc. Pacific 111, 1515-1523:.

Parameters

- **magB** (*float*, default=None) – The magnitude B of the star.
- **magV** (*float*, default=None) – The magnitude V of the star.
- **magK** (*float*, default=None) – The magnitude K of the star.

Note: If any of those values is 'None', 'nan' or higher than 49, it is not considered.

6.9 sora.stats

6.9.1 The Parameters Class

class sora.stats.**Parameters**

Creates a dictionary of Parameter objects. The structure of this class heavely borrows its structure from parameters.py (lmfit)

The dictionary contains all the parameters necessary to perform the fit of a model/objective function.

Parameters
dict (*dict*) – A dictionary of Parameter objects.

_addpar(*name*, *parameter*)

Private method to add the Parameter object to the Parameters dictionary.

Parameters

- **name** (*string*) – A name or label to describe the parameter
- **parameter** (*Parameter* object) – Parameter object containing the parameter values

Raises
ValueError – Error if passed object is not a valid Parameter object.

add(*name*, *value=None*, *minval=-inf*, *maxval=inf*, *free=True*, *std=None*, *initial_value=None*)

Include a Parameter values collection into the Parameters dictionary.

Parameters

- **name** (*str*) – A name or label to describe the parameter
- **value** (*float*, optional) – When first created it contains the initial estimate of the parameter. In a result object it contains the best fit value, by default None.
- **minval** (*float*, optional) – The lower bound used in the parameter variation, by default -inf
- **maxval** (*float*, optional) – The upper bound used in the parameter variation, by default inf
- **free** (*bool, optional*) – Defines if the parameter is allowed to vary, by default True
- **std** (*float*, *array*) – In the result object it contains the computed uncertainty.
- **initial_value** (*float*) – In the result object it contains the estimate before the fit.

Returns
object – A Parameters object containing the collection of parameters.

Return type
Parameters

`add_many`(**parlist*)

Add many parameters using a list of tuples.

Parameters
***parlist** (`list` of `tuple`) – A list of tuples containing the parameters in a sequence. The order in each tuple must follow the sequence: `(name, value, minval, maxval, free)`

Returns
object – A Parameters object containing the collection of parameters.

Return type
Parameters

`get_bounds`(*transposed=False*)

Method to get the bounds recorded in the Parameters object.

Parameters
transposed (*bool, optional*) – Returns the bounds in a transposed array, by default False.

Returns
array – Tuple array containing the bounds values.

Return type
tuple

`get_names`()

Method to get the names of each parameter in the object.

Returns
list – List of parameters names.

Return type
str, *list*

`get_uncertainties`()

Method to get the uncertainties stored in the Parameters object.

Returns
list – List of parameters uncertainties.

Return type
float, *list*

`get_values`()

Method to get the values stored in the Parameters object.

Returns
list – List of parameters values.

Return type
float

`get_varys`()

Method to get the FREE values stored in the Parameters object.

Returns
list – List of FREE parameters values.

Return type
tuple

`remove`(*name*)

Method to remove a Parameter from the Parameters dictionary.

Parameters
name (*str*) – The name of the parameter to be removed.

`valuesdict`()

Method returns a dictionary of parameter and values.

Returns
dict – Dictionary of parameters names and values.

Return type
dict

6.9.2 OptimizeResult Class

`class sora.stats.OptimizeResult`(***kwargs*)

An object that contains the results obtained by the fitting procedure

`params`

The best-fit parameters computed from the fit.

Type
Parameters object

`method`

Method used to compute the best fit solution.

Type
str

`var_names`

Ordered list of variable parameter names used in optimization, and useful for understanding the values in *initial_values* and *covar*.

Type
list

`covar`

Scaled covariance matrix from minimization, with rows and columns corresponding to *var_names*. The covariance matrix is obtained from the approximation of the inverse of the Hessian matrix. This covariance matrix is scaled by the reduced chisqr.

Type
numpy.ndarray

`initial_values`

Dictionary of initial values for varying parameters.

Type
numpy.ndarray

best_fit

List containing best fit values corresponding to *var_names*.

Type
numpy.ndarray

std

Array containing estimated N-sigma uncertainties, otherwise *inf*. Corresponding to *var_names*. In the *least_squares* case the formal numeric uncertainties are then derived from the covariance matrix. When uncertainties are unavailable Bootstraping can be used to derive the uncertainties. In this case confidence intervals (CI) limited by the provided sigma (z-score), e.g., 1-sigma CI will return the [0.16,0.84] quantiles as the low and high confidence intervals value.

Type
numpy.ndarray

bootstrap

If not None, an array containing the bootstraped solutions for each variable (col), by defalut None.

Type
None, *numpy.ndarray*

success

True if optimization algorithm converged.

Type
str

nvars

Number of free variables used in fit.

Type
int

ndata

Number of data points.

Type
int

dof

Degrees of freedom in fit (ndata - nvars).

Type
int

residual

Residual array of the objective function when using the parameters best-fit solution.

Type
numpy.ndarray

chisqr

A weighted sum of squared deviations. If the objective function does not provide weighed residuals then it only expresses the sum of squared deviations.

Type
float

redchi

The Reduced chi-square value: i is defined as chi-square per degree of freedom.

Type
float

scipy

When scipy module is used the *scipy.OptimizeResult* object is also returned.

Type
scipy.OptimizeResult object

emcee

Describe

Type
describe

Returns
object – A OptimizeResult object containing the collection of the results obtained by the fit.

Return type
OptimizeResult

summary()

Prints a summary of the results obtained by the fit contained in the OptimizeResult object.

CHAPTER

SEVEN

GETTING STARTED

[1]:
```
## SORA package
from sora import Occultation, Body, Star, LightCurve, Observer
from sora.prediction import prediction
from sora.extra import draw_ellipse

## Other main packages
from astropy.time import Time
import astropy.units as u

## Usual packages
import numpy as np
import matplotlib.pylab as pl
import os
```

```
SORA version: 1.0dev
```

Before analysing stellar occultations data, let's predict them.

To predict stellar occultation we needs the intended Solar System body ephemeris and a time window.

[2]:
```
# First, let's consider an Solar System Body

chariklo = Body(name='Chariklo',
                ephem=['guidelines/input/bsp/Chariklo.bsp', 'guidelines/input/bsp/de438_
↪small.bsp'])

print(chariklo)
```

```
Obtaining data for Chariklo from SBDB
################################################################################
                          10199 Chariklo (1997 CU26)
################################################################################
Object Orbital Class: Centaur
Spectral Type:
    SMASS: D  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Relatively featureless spectrum with very steep red slope.
Discovered 1997-Feb-15 by Spacewatch at Kitt Peak

Physical parameters:
Diameter:
    302 +/- 30 km
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),
```

(continues on next page)

(continued from previous page)

```
Rotation:
    7.004 +/- 0 h
    Reference: LCDB (Rev. 2021-June); Warner et al., 2009, [Result based on less than␣
↪full coverage, so that the period may be wrong by 30 percent or so.]  REFERENCE LIST:
↪[Fornasier, S.; Lazzaro, D.; Alvarez-Candal, A.; Snodgrass, C.; et al. (2014) Astron.␣
↪Astrophys. 568, L11.], [Leiva, R.; Sicardy, B.; Camargo, J.I.B.; Desmars, J.; et al.␣
↪(2017) Astron. J. 154, A159.]
Absolute Magnitude:
    6.58 +/- 0 mag
    Reference: MPO647128,
Albedo:
    0.045 +/- 0.01
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),

Ellipsoid: 151.0 x 151.0 x 151.0

----------- Ephemeris -----------

EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

[3]:
```
pred = prediction(body=chariklo, time_beg='2017-06-20',time_end='2017-06-27',mag_lim=16)

pred
```

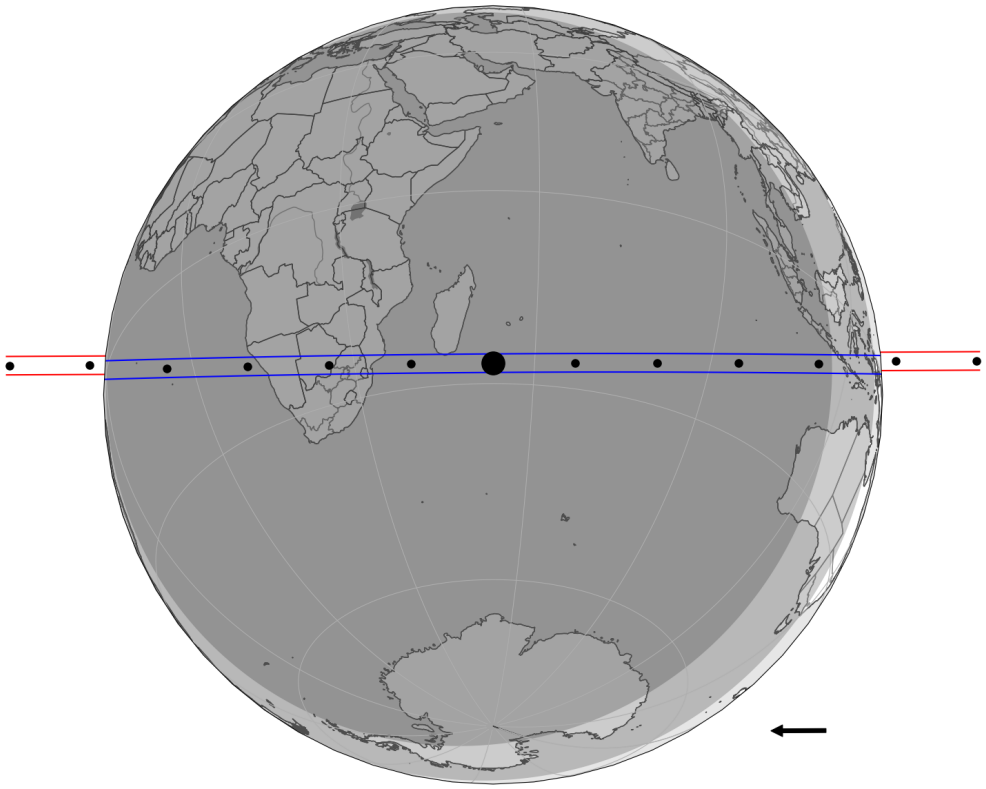
```
Ephemeris was split in 1 parts for better search of stars

Searching occultations in part 1/1
Generating Ephemeris between 2017-06-20 00:00:00.000 and 2017-06-26 23:59:00.000 ...
Downloading stars ...
    5 GaiaDR3 stars downloaded
Identifying occultations ...

2 occultations found.
```

[3]:
```
<PredictionTable length=2>
         Epoch             ICRS Star Coord at Epoch    ...  GaiaDR3 Source ID
                                                       ...
         object                      object             ...        str19
----------------------- ------------------------------ ... -------------------
2017-06-21 09:57:43.440 18 55 36.17454 -31 31 19.03261 ... 6760228702284187264
2017-06-22 21:18:48.200 18 55 15.65251 -31 31 21.67062 ... 6760223758801661440
```

[4]:
```
## ploting the occultation map

pred['2017-06-22 21:18'].plot_occ_map(nameimg='guidelines/figures/pred_map')
```

```
guidelines/figures/pred_map.png generated
```

```
  Object        Diam   Tmax   dots <> ra_offset_dec
10199 Chariklo  302 km   13.7s  60 s <>  +0.0   +0.0
```



```
 year-m-d    h:m:s UT     ra__dec__J2000__candidate    C/A    P/A    vel   Delta   G*  long
2017-06-22 21:18:48.200  18 55 15.6525 -31 31 21.671  0.049 359.72 -22.00  14.66  14.2   53
```

Now, let's start instantiating the Occultation

An occultation is defined by the occulting body, the occulted star, and the time of the occultation

[5]:
```
Occultation?
```

```
Init signature:
Occultation(
    star,
    body=None,
    ephem=None,
    time=None,
    reference_center='geocenter',
)
Docstring:
Instantiates the Occultation object and performs the reduction of the
occultation.

Attributes
----------
star : `sora.Star`, `str`, required
```

(continues on next page)

(continued from previous page)

```
    the coordinate of the star in the same reference frame as the ephemeris.
    It must be a Star object or a string with the coordinates of the object
    to search on Vizier.

body : `sora.Body`, `str`
    Object that will occult the star. It must be a Body object or its name
    to search in the Small Body Database.

ephem : `sora.Ephem`, `list`
    Object ephemeris. It must be an Ephemeris object or a list.

time : `str`, `astropy.time.Time`, required
    Reference time of the occultation. Time does not need to be exact, but
    needs to be within approximately 50 minutes of the occultation closest
    approach to calculate occultation parameters.

reference_center : `str`, `sora.Observer`, `sora.Spacecraft`
    A SORA observer object or a string 'geocenter'.
    The occultation parameters will be calculated in respect
    to this reference as center of projection.


Important
---------
When instantiating with "body" and "ephem", the user may define the
Occultation in 3 ways:

1. With `body` and `ephem`.

2. With only "body". In this case, the "body" parameter must be a Body
object and have an ephemeris associated (see Body documentation).

3. With only `ephem`. In this case, the `ephem` parameter must be one of the
Ephem Classes and have a name (see Ephem documentation) to search for the
body in the Small Body Database.
File:           ~/miniconda3/envs/sora-develop-stats/lib/python3.9/site-packages/sora_
↪astro-1.0.dev0-py3.9.egg/sora/occultation/core.py
Type:           type
Subclasses:
```

[6]:
```
star_occ = Star(coord='18 55 15.65250 -31 31 21.67051')
#star_occ = Star(code='6760223758801661440')

print(star_occ)
```

```
1 GaiaDR3 star found band={'G': 14.223702}
star coordinate at J2016.0: RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.
↪018 mas

Downloading star parameters from I/297/out
GaiaDR3 star Source ID: 6760223758801661440
ICRS star coordinate at J2016.0:
```

(continues on next page)

(continued from previous page)

```
RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.0180 mas
pmRA=3.556 +/- 0.025 mas/yr, pmDEC=-2.050 +/- 0.020 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.2121 +/- 0.0228 mas, Rad. Vel.=-40.49 +/- 3.73 km/s

Magnitudes: G: 14.224, B: 14.320, V: 13.530, R: 14.180, J: 12.395, H: 11.781,
            K: 11.627

Apparent diameter from Kervella et. al (2004):
    V: 0.0216 mas, B: 0.0216 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0238 mas, V: 0.0244 mas
    ms: B: 0.0261 mas, V: 0.0198 mas
    vs: B: 0.0350 mas, V: 0.0315 mas
```

[7]:
```
occ = Occultation(star=star_occ, body=chariklo, time='2017-06-22 21:18')

print(occ)
```

```
Stellar occultation of star GaiaDR3 6760223758801661440 by 10199 Chariklo (1997 CU26).

Geocentric Closest Approach: 0.049 arcsec
Instant of CA: 2017-06-22 21:18:48.200
Position Angle: 359.72 deg
Geocentric shadow velocity: -22.00 km / s
Sun-Geocenter-Target angle:  166.42 deg
Moon-Geocenter-Target angle: 149.11 deg


No observations reported

################################################################################
                                    STAR
################################################################################
GaiaDR3 star Source ID: 6760223758801661440
ICRS star coordinate at J2016.0:
RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.0180 mas
pmRA=3.556 +/- 0.025 mas/yr, pmDEC=-2.050 +/- 0.020 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.2121 +/- 0.0228 mas, Rad. Vel.=-40.49 +/- 3.73 km/s

Magnitudes: G: 14.224, B: 14.320, V: 13.530, R: 14.180, J: 12.395, H: 11.781,
            K: 11.627

Apparent diameter from Kervella et. al (2004):
    V: 0.0216 mas, B: 0.0216 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0238 mas, V: 0.0244 mas
    ms: B: 0.0261 mas, V: 0.0198 mas
    vs: B: 0.0350 mas, V: 0.0315 mas

Geocentric star coordinate at occultation Epoch (2017-06-22 21:18:48.200):
```

(continues on next page)

(continued from previous page)

```
RA=18h55m15.65251s +/- 0.0323 mas, DEC=-31d31m21.6706s +/- 0.0341 mas

###############################################################################
                          10199 Chariklo (1997 CU26)
###############################################################################
Object Orbital Class: Centaur
Spectral Type:
    SMASS: D  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Relatively featureless spectrum with very steep red slope.
Discovered 1997-Feb-15 by Spacewatch at Kitt Peak

Physical parameters:
Diameter:
    302 +/- 30 km
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),
Rotation:
    7.004 +/- 0 h
    Reference: LCDB (Rev. 2021-June); Warner et al., 2009, [Result based on less than␣
↪full coverage, so that the period may be wrong by 30 percent or so.]  REFERENCE LIST:
↪[Fornasier, S.; Lazzaro, D.; Alvarez-Candal, A.; Snodgrass, C.; et al. (2014) Astron.␣
↪Astrophys. 568, L11.], [Leiva, R.; Sicardy, B.; Camargo, J.I.B.; Desmars, J.; et al.␣
↪(2017) Astron. J. 154, A159.]
Absolute Magnitude:
    6.58 +/- 0 mag
    Reference: MPO647128,
Albedo:
    0.045 +/- 0.01
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),

Ellipsoid: 151.0 x 151.0 x 151.0

----------- Ephemeris -----------

EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

After that, we instantiate the observers and their light curves

Observers

Now let's define our observers, they can be setted manually or from the MPC database

[8]:
```
### User

out = Observer(name='Outeniqua'  ,lon='+16 49 17.710', lat='-21 17 58.170', height =1416)
ond = Observer(name='Onduruquea' ,lon='+15 59 33.750', lat='-21 36 26.040', height =1220)
tiv = Observer(name='Tivoli'     ,lon='+18 01 01.240', lat='-23 27 40.190', height =1344)
whc = Observer(name='Windhoek'   ,lon='+17 06 31.900', lat='-22 41 55.160', height =1902)
hak = Observer(name='Hakos'      ,lon='+16 21 41.320', lat='-23 14 11.040', height =1843)
```

(continues on next page)

(continued from previous page)

```
print(tiv)

print('\n')

### MPC Database Search

opd = Observer(name='Observatorio Pico dos Dias',code='874')

print(opd)
```

```
Site: Tivoli
Geodetic coordinates: Lon: 18d01m01.24s, Lat: -23d27m40.19s, height: 1.344 km


Site: Observatorio Pico dos Dias
Geodetic coordinates: Lon: -45d34m57.54s, Lat: -22d32m07.74756091s, height: 1.811 km
```

Light Curves

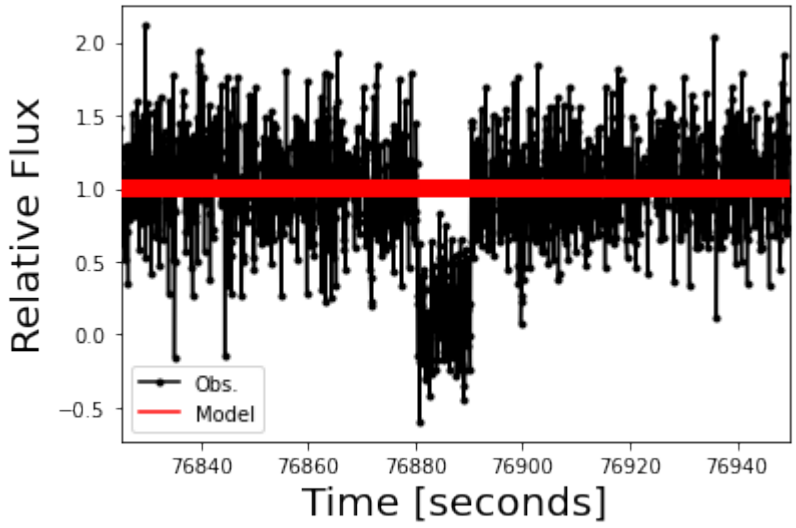
Now let's define our light curves, they can be instanciated from different way: - **(i)** Manually with arrays containing the flux and the times; - **(ii)** Read an ASCII file; - **(iii)** Already obtained times.

Outeniqua (Namibia)

[9]:
```
out_lc = LightCurve(name='Outeniqua lc',file='guidelines/input/lightcurves/lc_example.dat
↪',
                    exptime=0.100, usecols=[0,1])

print(out_lc)
```

```
Light curve name: Outeniqua lc
Initial time: 2017-06-22 21:20:00.056 UTC
End time:     2017-06-22 21:23:19.958 UTC
Duration:     3.332 minutes
Time offset:  0.000 seconds

Exposure time:    0.1000 seconds
Cycle time:       0.1002 seconds
Num. data points: 2000


There is no occultation associated with this light curve.

Object LightCurve model was not fitted.

Immersion and emersion times were not fitted or instantiated.
```

[10]:
```
out_lc.plot_lc()
pl.xlim(76825,76950)
pl.show()
```



The light curve occultation model considers some physical parameters from the event: - Distance between the geocenter and the occulting object (AU); - Star diameter at the occulting object's distance (km); - Nominal Velocity of the event (km/s);

These parameters can be automatic calculated as we connect the `LightCurve` and the `Observer` to the `Ocultation` Object.

[11]:
```
occ.chords.add_chord(observer=out,lightcurve=out_lc)

print(out_lc)
```

```
Light curve name: Outeniqua lc
Initial time: 2017-06-22 21:20:00.056 UTC
End time:     2017-06-22 21:23:19.958 UTC
Duration:     3.332 minutes
Time offset:  0.000 seconds

Exposure time:    0.1000 seconds
Cycle time:       0.1002 seconds
Num. data points: 2000

Bandpass:              0.700 +/- 0.300 microns
Object Distance:       14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:         0.040 seconds or 0.87 km
Stellar size effect:   0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion and emersion times were not fitted or instantiated.
```

```
/home/rcboufleur/miniconda3/envs/sora-develop-stats/lib/python3.9/site-packages/sora_
↪astro-1.0.dev0-py3.9.egg/sora/body/core.py:332: UserWarning: H and/or G is not defined␣
↪for 10199 Chariklo. Searching into JPL Horizons service
```

Now, appart from the `LightCurve` Object having the needed parameters, also the `Occultation` object can acess the information from this `Chord`.

[12]:
```
print(occ.chords)
```

```
--------------------------------------------------------------------------------
Site: Outeniqua
Geodetic coordinates: Lon: 16d49m17.71s, Lat: -21d17m58.17s, height: 1.416 km
Target altitude: 56.7 deg
Target azimuth:  115.3 deg

Light curve name: Outeniqua lc
Initial time: 2017-06-22 21:20:00.056 UTC
End time:     2017-06-22 21:23:19.958 UTC
Duration:     3.332 minutes
Time offset:  0.000 seconds

Exposure time:    0.1000 seconds
Cycle time:       0.1002 seconds
Num. data points: 2000

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion and emersion times were not fitted or instantiated.
```

[13]:
```
## We fit the modelled light curve, using chi square minimization and Monte Carlo␣
↪procedures

out_lc.occ_lcfit?
```

```
Signature: out_lc.occ_lcfit(**kwargs)
Docstring:
Monte Carlo chi square fit for occultations lightcurve.

Parameters
----------
tmin : `int`, `float`
    Minimum time to consider in the fit procedure, in seconds.

tmax : `int`, `float`
    Maximum time to consider in the fit procedure, in seconds.

flux_min : `int`, `float`, default=0
    Bottom flux (only object).

flux_max :`int`, `float`, default=1
```

(continues on next page)

(continued from previous page)

```
    Base flux (object plus star).

immersion_time : `int`, `float`
    Initial guess for immersion time, in seconds.

emersion_time : `int`, `float`
    Initial guess for emersion time, in seconds.

opacity : `int`, `float`, default=1
    Initial guess for opacity. Opaque = 1, Transparent = 0.

delta_t : `int`, `float`
    Interval to fit immersion or emersion time.

dopacity : `int`, `float`, default=0
    Interval to fit opacity.

sigma : `int`, `float`, `array`, 'auto'
    Fluxes errors. If None it will use the `self.dflux`. If 'auto' it
    will calculate using the region outside the event.

loop : `int`, default=10000
    Number of tests to be done.

sigma_result : `int`, `float`
    Sigma value to be considered as result.

method : `str`, default=`chisqr`
    Method used to perform the fit. Available methods are:
    `chisqr` : monte carlo computation method used in versions of SORA <= 0.2.1.
    `fastchi` : monte carlo computation method, allows multithreading.
    `least_squares` or `ls`: best fit done used levenberg marquardt convergence␣
↪algorithm.
    `differential_evolution` or `de`: best fit done using genetic algorithms.
    All methods return a Chisquare object.

threads : `int`
    Number of threads/workers used to perform parallel computations of the chi square
    object. It works with all methods except `chisqr`, by default 1.

Returns
-------
chi2 : `sora.extra.ChiSquare`
    ChiSquare object.
File:      ~/miniconda3/envs/sora-develop-stats/lib/python3.9/site-packages/sora_astro-1.
↪0.dev0-py3.9.egg/sora/lightcurve/core.py
Type:      method
```

[14]:
```
## An automatic version can be used for cases where the occultation is obvious!!
## This process may take some minutes to run!!

out_chi2 = out_lc.occ_lcfit(loop=1000)
```

(continues on next page)

(continued from previous page)

```
print('\n')
print(out_chi2)
```

```
LightCurve fit: ||  - 100%


Minimum chi-square: 473.931
Number of fitted points: 496
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 0.959

immersion:
    1-sigma: 76880.322 +/- 0.032
    3-sigma: 76880.353 +/- 0.131

emersion:
    1-sigma: 76890.349 +/- 0.031
    3-sigma: 76890.347 +/- 0.108
```

[15]:
```
## However, we believe that the user should set the parameters by hand!!
## The complete description of each parameter can be seen at the function Docstring.
## This process may take some minutes to run!!

out_chi2 = out_lc.occ_lcfit(tmin=76875.0, tmax=76895.0,
                            immersion_time=76880.3,
                            emersion_time=76890.3,
                            delta_t=0.2, loop=10000)
print('\n')
print(out_chi2)
```

```
LightCurve fit: ||  - 100%


Minimum chi-square: 192.774
Number of fitted points: 200
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 0.974

immersion:
    1-sigma: 76880.325 +/- 0.027
    3-sigma: 76880.347 +/- 0.121

emersion:
    1-sigma: 76890.351 +/- 0.029
    3-sigma: 76890.348 +/- 0.101
```

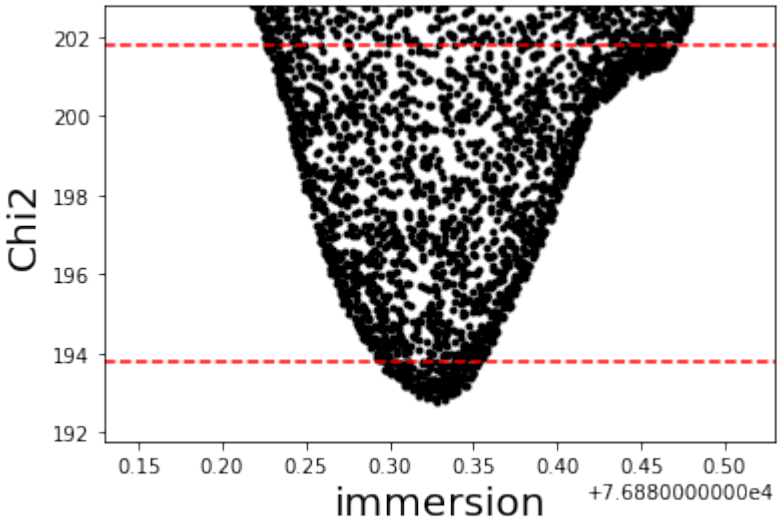
The user can visually acess the quality of the fit by ploting the `ChiSquare` object.

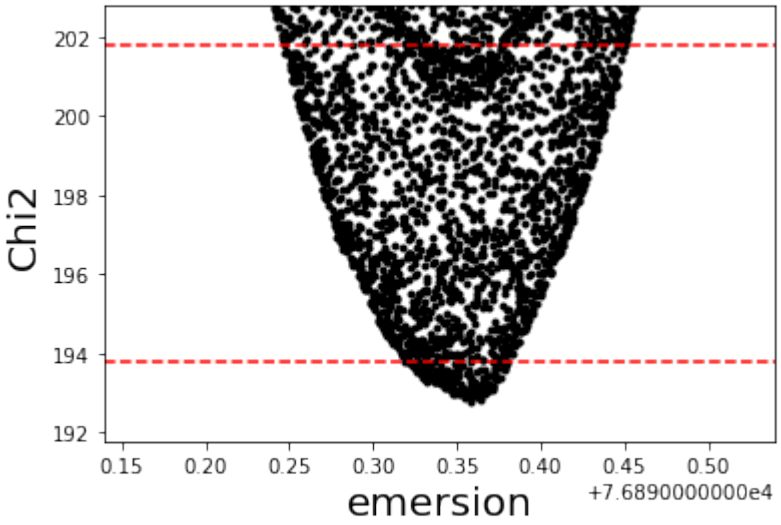
[16]:
```
out_chi2.plot_chi2('immersion')
pl.xlim(76880.33 - 0.20, 76880.33 + 0.20)
pl.show()
```

(continues on next page)

(continued from previous page)

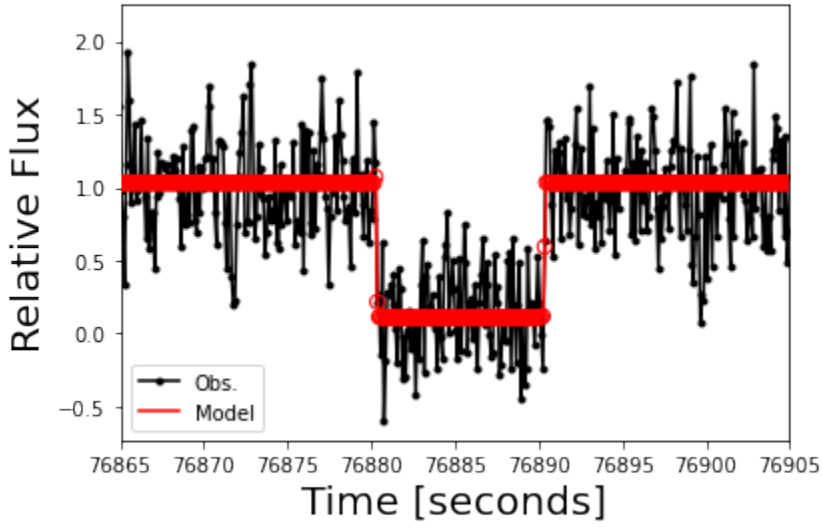
```
out_chi2.plot_chi2('emersion')
pl.xlim(76890.34 - 0.20, 76890.34 + 0.20)
pl.show()
```

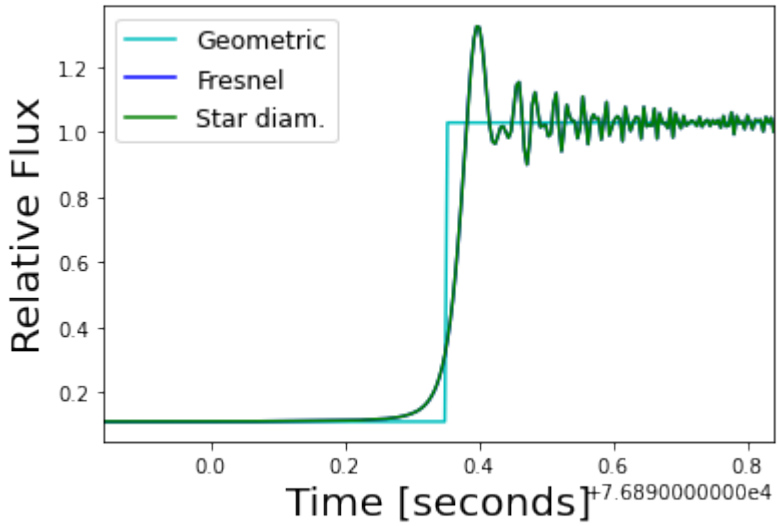




Also, the user can visually acess the quality of the fit by ploting the `LightCurve`.

[17]:
```
out_lc.plot_lc()
pl.xlim(76865, 76905)
pl.show()

out_lc.plot_model()
pl.xlim(76890.34-0.5, 76890.34+0.5)
pl.legend(ncol=1, fontsize=12.5, loc=2)
pl.show()
```





[18]:
```
print(out_lc)
```

```
Light curve name: Outeniqua lc
Initial time: 2017-06-22 21:20:00.056 UTC
End time:     2017-06-22 21:23:19.958 UTC
Duration:     3.332 minutes
Time offset:  0.000 seconds

Exposure time:    0.1000 seconds
Cycle time:       0.1002 seconds
Num. data points: 2000

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km
```

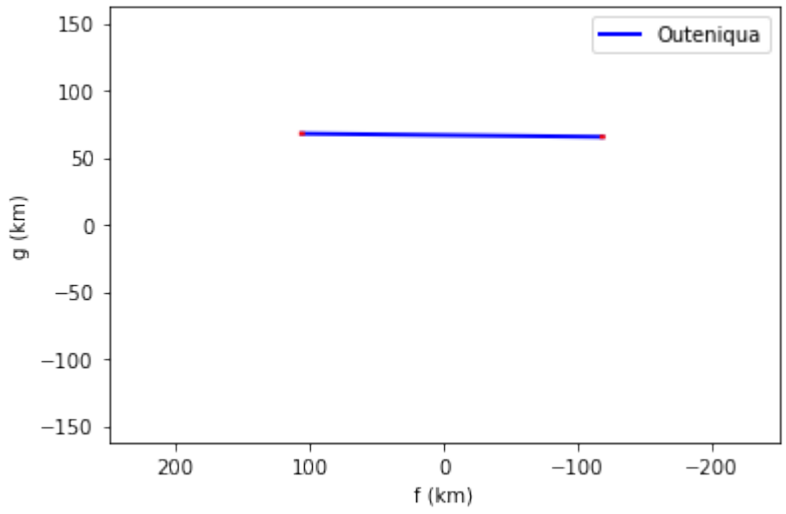
(continues on next page)

(continued from previous page)

```
Inst. response:       0.100 seconds or 2.20 km
Dead time effect:     0.000 seconds or 0.00 km
Model resolution:     0.004 seconds or 0.09 km
Modelled baseflux:    1.029
Modelled bottomflux:  0.109
Light curve sigma:    0.307

Immersion time: 2017-06-22 21:21:20.325 UTC +/- 0.027 seconds
Emersion time:  2017-06-22 21:21:30.351 UTC +/- 0.029 seconds

Monte Carlo chi square fit.

Minimum chi-square: 192.774
Number of fitted points: 200
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 0.974

immersion:
    1-sigma: 76880.325 +/- 0.027
    3-sigma: 76880.347 +/- 0.121

emersion:
    1-sigma: 76890.351 +/- 0.029
    3-sigma: 76890.348 +/- 0.101
```

Finally, the user can visually see the chord in the Sky-plane using the `Chord` Object.

[19]:
```
occ.chords.plot_chords(segment='positive', color='blue')
occ.chords.plot_chords(segment='error', color='red')
pl.legend()
pl.xlim(+250,-250)
pl.ylim(-250,+250)
pl.show()
```



Now, let's add the other chords of this occultation.

Onduruquea (Namibia)

[20]:
```
ond_lc = LightCurve(name='Onduruquea lc',
                    initial_time='2017-06-22 21:11:52.175',
                    end_time ='2017-06-22 21:25:13.389',
                    immersion='2017-06-22 21:21:22.213', immersion_err=0.010,
                    emersion ='2017-06-22 21:21:33.824', emersion_err=0.011)

occ.chords.add_chord(observer=ond,lightcurve=ond_lc)
```

[20]:
```
<Chord: Onduruquea>
```

[21]:
```
print(occ.chords['Onduruquea'])
```

```
--------------------------------------------------------------------------------
Site: Onduruquea
Geodetic coordinates: Lon: 15d59m33.75s, Lat: -21d36m26.04s, height: 1.220 km
Target altitude: 56.1 deg
Target azimuth:  114.7 deg

Light curve name: Onduruquea lc
Initial time: 2017-06-22 21:11:52.175 UTC
End time:     2017-06-22 21:25:13.389 UTC
Duration:     13.354 minutes
Time offset:  0.000 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:              0.700 +/- 0.300 microns
Object Distance:       14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:         0.040 seconds or 0.87 km
Stellar size effect:   0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:22.213 UTC +/- 0.010 seconds
Emersion time:  2017-06-22 21:21:33.824 UTC +/- 0.011 seconds
```

Tivoli (Namibia)

[22]:
```
tiv_lc = LightCurve(name='Tivoli lc',
                    initial_time='2017-06-22 21:16:00.094',
                    end_time ='2017-06-22 21:28:00.018',
                    immersion='2017-06-22 21:21:15.628',immersion_err=0.011,
                    emersion ='2017-06-22 21:21:19.988',emersion_err=0.038)

occ.chords.add_chord(observer=tiv, lightcurve=tiv_lc)
```

[22]:
```
<Chord: Tivoli>
```

Windhoek (Namibia)

When there is two chords at the same stations is important to define their names as different values

[23]:
```
## C14
whc_c14_lc = LightCurve(name='Windhoek C14 lc',
                    initial_time='2017-06-22 21:12:48.250',
                    end_time ='2017-06-22 21:32:47.963',
                    immersion='2017-06-22 21:21:17.609',immersion_err=0.024,
                    emersion ='2017-06-22 21:21:27.564',emersion_err=0.026)

occ.chords.add_chord(name='Windhoek C14 lc', observer=whc, lightcurve=whc_c14_lc)

## D16
whc_d16_lc = LightCurve(name='Windhoek D16 lc',
                    initial_time='2017-06-22 21:20:01.884',
                    end_time ='2017-06-22 21:22:21.894',
                    immersion='2017-06-22 21:21:17.288',immersion_err=0.028,
                    emersion ='2017-06-22 21:21:27.228',emersion_err=0.034)

occ.chords.add_chord(name='Windhoek D16 lc', observer=whc, lightcurve=whc_d16_lc)
```

[23]:
```
<Chord: Windhoek D16 lc>
```

Hakos (Namibia)

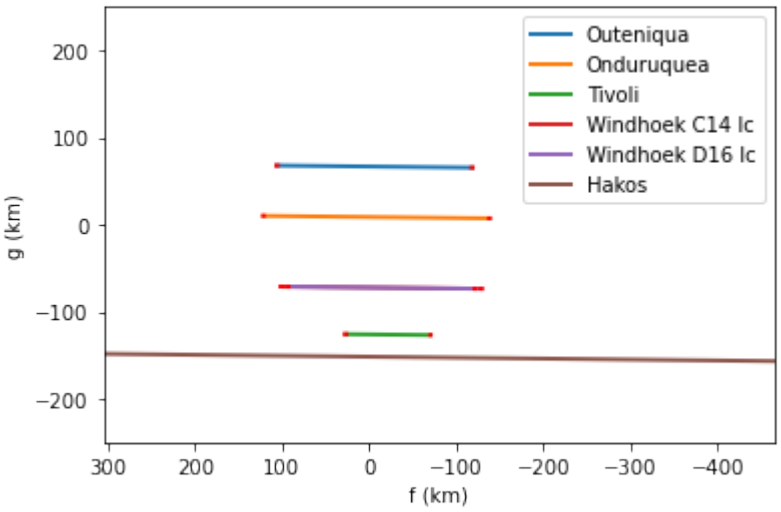
[24]:
```
#Also negatives chords can be added
hak_lc = LightCurve(name='Hakos lc',
                    initial_time='2017-06-22 21:10:19.461',
                    end_time ='2017-06-22 21:30:19.345')

occ.chords.add_chord(observer=hak, lightcurve=hak_lc)
```

[24]:
```
<Chord: Hakos>
```

Chords display and ellipse fit

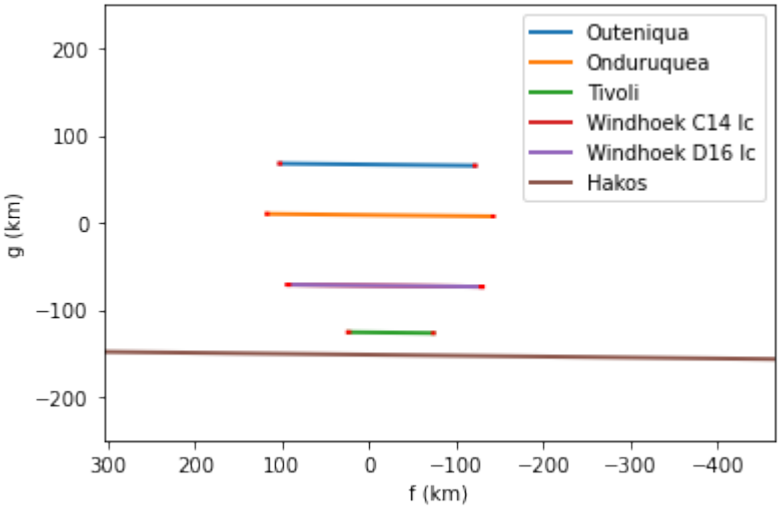
After all light curves were instanciated and/or fitted, the next step is to plot the chords and fit the elipse.

[25]:
```
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```



[26]:
```
## We can add known time offsets due to camera features

out_lc.dt = -0.150
ond_lc.dt = -0.190
tiv_lc.dt = -0.150
whc_c14_lc.dt = -0.375
whc_d16_lc.dt = +0.000
hak_lc.dt = -0.200
```

[27]:
```
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```



The next step is to fit an ellipse to the chords

[28]:
```
## We fit a ellipse using chi square minimization and Monte Carlo procedures, the
## The complete description of each parameter can be seen at the function Docstring.

occ.fit_ellipse?
```

```
Signature: occ.fit_ellipse(**kwargs)
Docstring:
Fits an ellipse to given occultation using given parameters.

Parameters
----------
center_f : `int`, `float`, default=0
    The coordinate in f of the ellipse center.

center_g : `int`, `float`, default=0
    The coordinate in g of the ellipse center.

equatorial_radius : `int`, `float`
    The Equatorial radius (semi-major axis) of the ellipse.

oblateness : `int`, `float`, default=0
    The oblateness of the ellipse.

position_angle : `int`, `float`, default=0
    The pole position angle of the ellipse in degrees.
    Zero is in the North direction ('g-positive'). Positive clockwise.

dcenter_f : `int`, `float`
    Interval for coordinate f of the ellipse center.

dcenter_g : `int`, `float`
    Interval for coordinate g of the ellipse center.

dequatorial_radius `int`, `float`
    Interval for the Equatorial radius (semi-major axis) of the ellipse.

doblateness : `int`, `float`
    Interval for the oblateness of the ellipse

dposition_angle : `int`, `float`
    Interval for the pole position angle of the ellipse in degrees.

loop : `int`, default=10000000
    The number of ellipses to attempt fitting.

dchi_min : `int`, `float`
    If given, it will only save ellipsis which chi square are smaller than
    chi_min + dchi_min. By default `None` when used with `method='chisqr`, and
    `3` for other methods.

number_chi : `int`, default=10000
    In the `chisqr` method if `dchi_min` is given, the procedure is repeated until
    `number_chi` is reached.
```

(continues on next page)

(continued from previous page)

```
    In other methods it is the number of values (simulations) that should lie within
    the provided `sigma_result`.

verbose : `bool`, default=False
    If True, it prints information while fitting.

ellipse_error : `int`, `float`
    Model uncertainty to be considered in the fit, in km.

sigma_result : `int`, `float`
    Sigma value to be considered as result.

method : `str`, default=`least_squares`
    Method used to perform the fit. Available methods are:
    `chisqr` : monte carlo computation method used in versions of SORA <= 0.2.1.
    `fastchi` : monte carlo computation method, allows multithreading .
    `least_squares` or `ls`: best fit done used levenberg marquardt convergence␣
↪algorithm.
    `differential_evolution` or ``de`: best fit done using genetic algorithms.
    All methods return a Chisquare object.

threads : `int`
    Number of threads/workers used to perform parallel computations of the chi square
    object. It works with all methods except `chisqr`, by default 1.

Returns
-------
chisquare : `sora.ChiSquare`
    A ChiSquare object with all parameters.

Important
---------
Each occultation is added as the first argument(s) directly.

Mandatory input parameters: 'center_f', 'center_g', 'equatorial_radius',
'oblateness', and 'position_angle'.

Parameters fitting interval: 'dcenter_f', 'dcenter_g', 'dequatorial_radius',
'doblateness', and 'dposition_angle'. Default values are set to zero.
Search done between (value - dvalue) and (value + dvalue).


Examples
--------
To fit the ellipse to the chords of occ1 Occultation object:

>>> fit_ellipse(occ1, **kwargs)

To fit the ellipse to the chords of occ1 and occ2 Occultation objects together:

>>> fit_ellipse(occ1, occ2, **kwargs)
File:      ~/miniconda3/envs/sora-develop-stats/lib/python3.9/site-packages/sora_astro-1.
```

(continues on next page)

(continued from previous page)

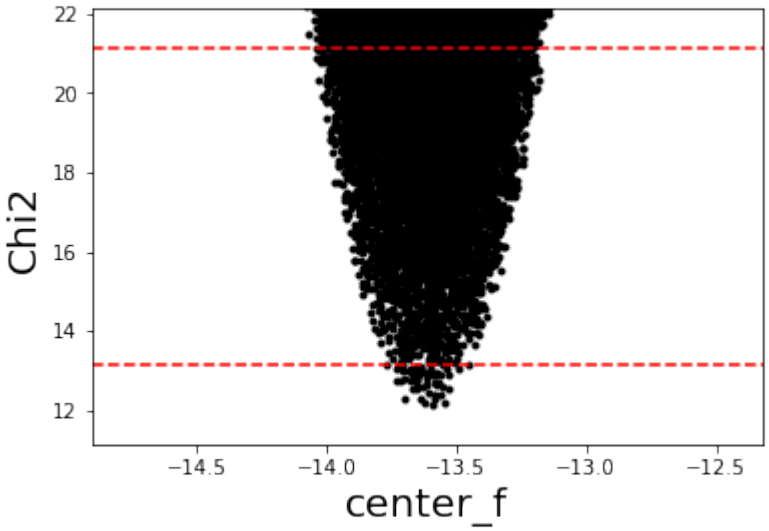
```
↪0.dev0-py3.9.egg/sora/occultation/core.py
Type:      method
```

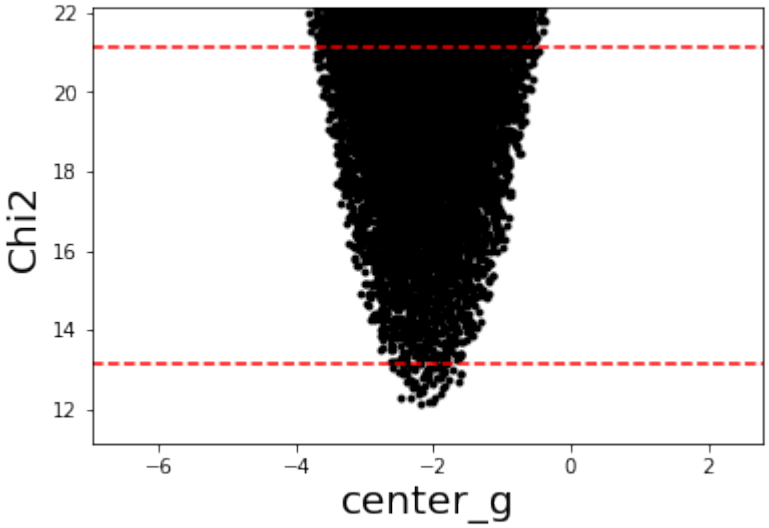
[29]:
```
### This may take some minutes to run!!

ellipse_chi2  = occ.fit_ellipse(center_f=-15.046, center_g=-2.495, dcenter_f=3, dcenter_
↪g=10,
                                equatorial_radius=138, dequatorial_radius=3,␣
↪oblateness=0.093,
                                doblateness=0.02, position_angle=126, dposition_angle=10␣
↪,loop=10000000,
                                dchi_min=10,number_chi=10000)


print(ellipse_chi2)
```

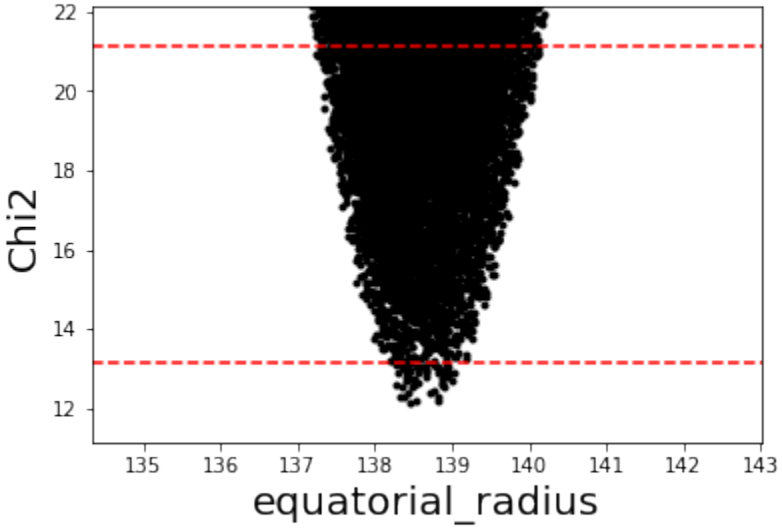
```
Minimum chi-square: 12.130
Number of fitted points: 10
Number of fitted parameters: 5
Minimum chi-square per degree of freedom: 2.426

center_f:
    1-sigma: -13.613 +/- 0.120
    3-sigma: -13.611 +/- 0.430

center_g:
    1-sigma: -2.094 +/- 0.499
    3-sigma: -2.089 +/- 1.626

equatorial_radius:
    1-sigma: 138.657 +/- 0.373
    3-sigma: 138.673 +/- 1.445

oblateness:
    1-sigma: 0.086 +/- 0.003
    3-sigma: 0.086 +/- 0.010

position_angle:
    1-sigma: 123.956 +/- 1.496
    3-sigma: 124.121 +/- 5.140
```

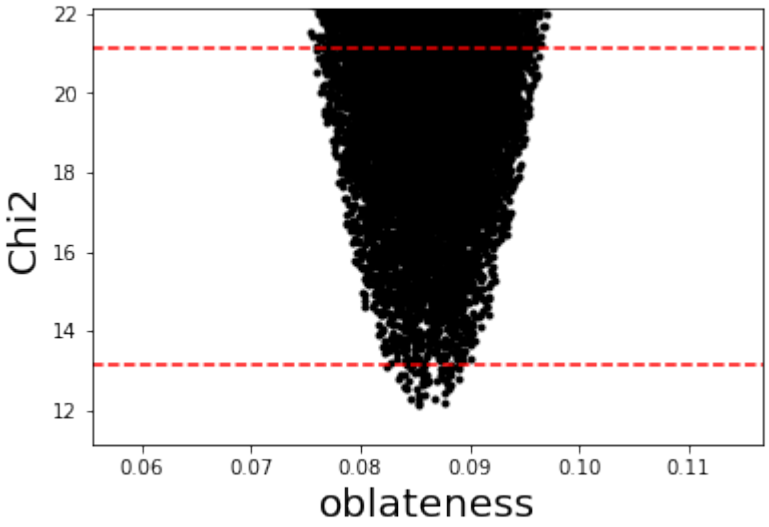
Similar, to the `LightCurve` fit, the user can visually acess the quality of the fit by ploting the `ChiSquare` object.

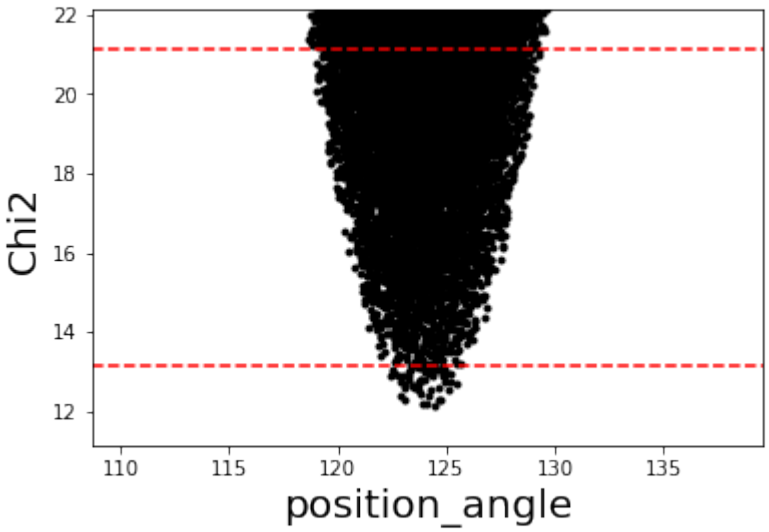
[30]:
```
ellipse_chi2.plot_chi2()
```


Chi2
22
20
18
16
14
12
−14.5
−14.0
−13.5
−13.0
−12.5
center_f


Chi2
22
20
18
16
14
12
−6
−4
−2
0
2
center_g

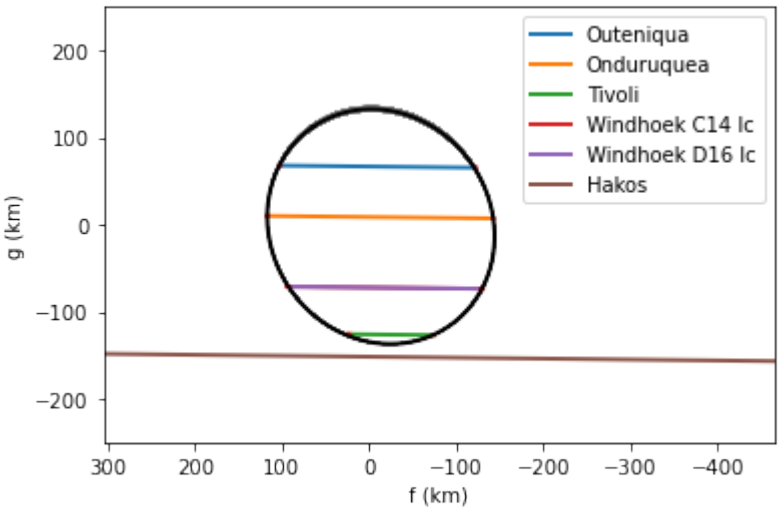

Chi2
22
20
18
16
14
12
135
136
137
138
139
140
141
142
143
equatorial_radius





Also, the user can visually acess the quality of the fit by ploting the `Chords` and the fitted ellipses.

[31]:
```
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

#plotting the best fitted ellipse, in black
draw_ellipse(**ellipse_chi2.get_values())

# ploting all the ellipses within 3-sigma, in gray
draw_ellipse(**ellipse_chi2.get_values(sigma=3),alpha=1.0)

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```



The resulting values can be acessed from the Dictionaries `Occultation.fitted_params` **and** `Occultation.chi2_params`

[32]:
```
occ.fitted_params
```

[32]:
```
{'equatorial_radius': [138.65650796361007, 0.3729805506807651],
 'center_f': [-13.612927096827903, 0.11996518649955146],
 'center_g': [-2.0936385736495167, 0.4992405703706848],
 'oblateness': [0.08573692161473534, 0.003305360393147015],
 'position_angle': [123.95618703950413, 1.4957816035600828]}
```

[33]:
```
occ.chi2_params
```

[33]:
```
{'chord_name': ['Outeniqua_immersion',
  'Outeniqua_emersion',
  'Onduruquea_immersion',
  'Onduruquea_emersion',
  'Tivoli_immersion',
  'Tivoli_emersion',
  'Windhoek C14 lc_immersion',
  'Windhoek C14 lc_emersion',
  'Windhoek D16 lc_immersion',
  'Windhoek D16 lc_emersion'],
 'radial_dispersion': array([ 0.64886238, -0.82234285, -0.20941195,  0.0507056 ,  0.
↪14404744,
       -2.02271261,  0.3135147 , -0.1926379 , -0.76516855,  0.53220478]),
 'position_angle': array([301.95754708,  58.93979955, 274.10063776,  84.70868609,
       205.80484092, 163.27224646, 238.45346645, 123.09039914,
       238.19141651, 122.87262257]),
 'radial_error': array([0.60995255, 0.64472971, 0.22353256, 0.24588816, 0.24592892,
       0.8495755 , 0.53652931, 0.58124429, 0.62595044, 0.76008871]),
 'chi2_min': 12.129614721820191,
 'nparam': 5,
 'npts': 10}
```

Besides the size and shape of the body the astrometrical positions obtained using stellar occultation is also a relevant result from the occultation and it has a precision that can be compared with space probes results (few

km)

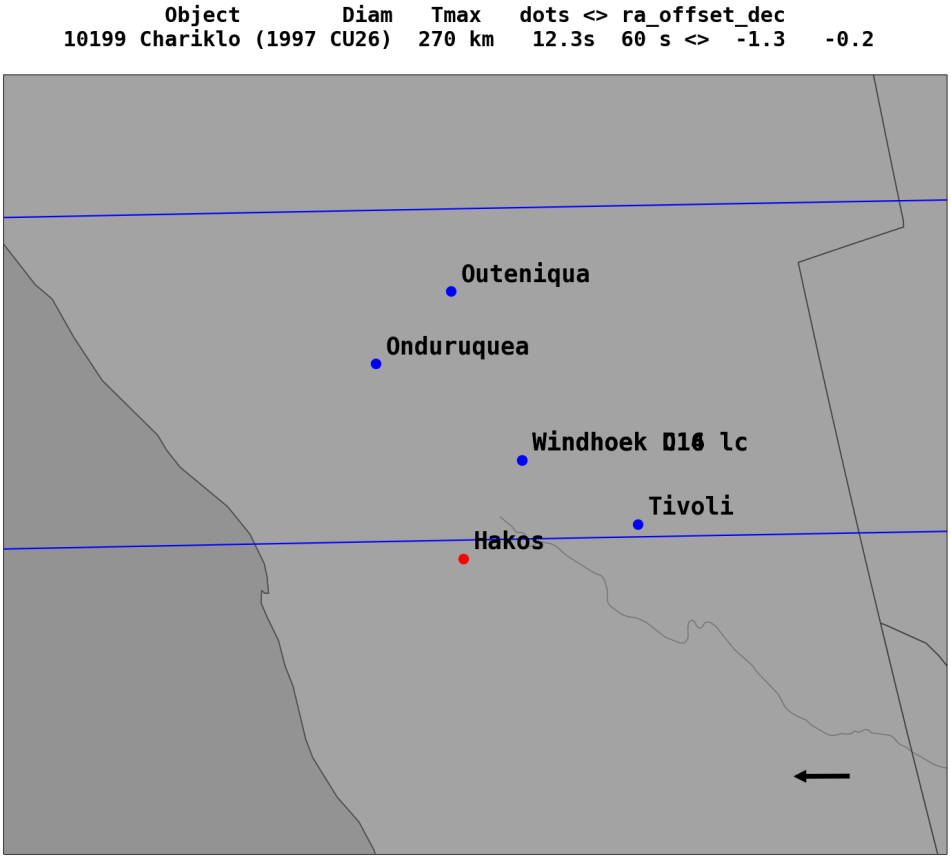
[34]:
```
occ.new_astrometric_position()
```

```
Ephemeris offset (km): X = -13.6 km +/- 0.1 km; Y = -2.1 km +/- 0.5 km
Ephemeris offset (mas): da_cos_dec = -1.280 +/- 0.011; d_dec = -0.197 +/- 0.047

Astrometric object position at time 2017-06-22 21:18:48.200 for reference 'geocenter'
RA = 18 55 15.6523911 +/- 0.034 mas; DEC = -31 31 21.622094 +/- 0.058 mas
```

After the instanciation of the `Chords` **and the ellipse fit, the posfit occultation map can be plotted.**

[35]:
```
occ.plot_occ_map(centermap_delta=[-3500,+400],zoom=20,nameimg='guidelines/figures/map_
↪posfit')
```

```
Projected shadow radius = 135.0 km
guidelines/figures/map_posfit.png generated
```



Finally, the log contains all the details

[36]:
```
print(occ)
```

```
Stellar occultation of star GaiaDR3 6760223758801661440 by 10199 Chariklo (1997 CU26).

Geocentric Closest Approach: 0.049 arcsec
Instant of CA: 2017-06-22 21:18:48.200
Position Angle: 359.72 deg
Geocentric shadow velocity: -22.00 km / s
Sun-Geocenter-Target angle:  166.42 deg
Moon-Geocenter-Target angle: 149.11 deg


5 positive observations
1 negative observations

################################################################################
                                      STAR
################################################################################
GaiaDR3 star Source ID: 6760223758801661440
ICRS star coordinate at J2016.0:
RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.0180 mas
pmRA=3.556 +/- 0.025 mas/yr, pmDEC=-2.050 +/- 0.020 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.2121 +/- 0.0228 mas, Rad. Vel.=-40.49 +/- 3.73 km/s

Magnitudes: G: 14.224, B: 14.320, V: 13.530, R: 14.180, J: 12.395, H: 11.781,
            K: 11.627

Apparent diameter from Kervella et. al (2004):
    V: 0.0216 mas, B: 0.0216 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0238 mas, V: 0.0244 mas
    ms: B: 0.0261 mas, V: 0.0198 mas
    vs: B: 0.0350 mas, V: 0.0315 mas

Geocentric star coordinate at occultation Epoch (2017-06-22 21:18:48.200):
RA=18h55m15.65251s +/- 0.0323 mas, DEC=-31d31m21.6706s +/- 0.0341 mas

################################################################################
                           10199 Chariklo (1997 CU26)
################################################################################
Object Orbital Class: Centaur
Spectral Type:
    SMASS: D  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Relatively featureless spectrum with very steep red slope.
Discovered 1997-Feb-15 by Spacewatch at Kitt Peak

Physical parameters:
Diameter:
    302 +/- 30 km
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),
Rotation:
    7.004 +/- 0 h
    Reference: LCDB (Rev. 2021-June); Warner et al., 2009, [Result based on less than␣
↪full coverage, so that the period may be wrong by 30 percent or so.]  REFERENCE LIST:
```

(continues on next page)

(continued from previous page)

```
↪[Fornasier, S.; Lazzaro, D.; Alvarez-Candal, A.; Snodgrass, C.; et al. (2014) Astron.␣
↪Astrophys. 568, L11.], [Leiva, R.; Sicardy, B.; Camargo, J.I.B.; Desmars, J.; et al.␣
↪(2017) Astron. J. 154, A159.]
Absolute Magnitude:
    6.58 +/- 0 mag
    Reference: JPL Horizons,
Phase Slope:
    0.15 +/- 0
    Reference: JPL Horizons,
Albedo:
    0.045 +/- 0.01
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),

Ellipsoid: 151.0 x 151.0 x 151.0

----------- Ephemeris -----------

EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec


################################################################################
                              POSITIVE OBSERVATIONS
################################################################################

--------------------------------------------------------------------------------
Site: Outeniqua
Geodetic coordinates: Lon: 16d49m17.71s, Lat: -21d17m58.17s, height: 1.416 km
Target altitude: 56.6 deg
Target azimuth:  115.3 deg

Light curve name: Outeniqua lc
Initial time: 2017-06-22 21:20:00.056 UTC
End time:     2017-06-22 21:23:19.958 UTC
Duration:     3.332 minutes
Time offset:  -0.150 seconds

Exposure time:    0.1000 seconds
Cycle time:       0.1002 seconds
Num. data points: 2000

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km
Inst. response:       0.100 seconds or 2.20 km
Dead time effect:     0.000 seconds or 0.00 km
Model resolution:     0.004 seconds or 0.09 km
Modelled baseflux:    1.029
Modelled bottomflux:  0.109
```

(continues on next page)

(continued from previous page)

```
Light curve sigma:    0.307

Immersion time: 2017-06-22 21:21:20.175 UTC +/- 0.027 seconds
Emersion time:  2017-06-22 21:21:30.201 UTC +/- 0.029 seconds

Monte Carlo chi square fit.

Minimum chi-square: 192.774
Number of fitted points: 200
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 0.974

immersion:
    1-sigma: 76880.325 +/- 0.027
    3-sigma: 76880.347 +/- 0.121

emersion:
    1-sigma: 76890.351 +/- 0.029
    3-sigma: 76890.348 +/- 0.101


--------------------------------------------------------------------------------
Site: Onduruquea
Geodetic coordinates: Lon: 15d59m33.75s, Lat: -21d36m26.04s, height: 1.220 km
Target altitude: 56.1 deg
Target azimuth:  114.7 deg

Light curve name: Onduruquea lc
Initial time: 2017-06-22 21:11:52.175 UTC
End time:     2017-06-22 21:25:13.389 UTC
Duration:     13.354 minutes
Time offset:  -0.190 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:22.023 UTC +/- 0.010 seconds
Emersion time:  2017-06-22 21:21:33.634 UTC +/- 0.011 seconds


--------------------------------------------------------------------------------
Site: Tivoli
Geodetic coordinates: Lon: 18d01m01.24s, Lat: -23d27m40.19s, height: 1.344 km
Target altitude: 58.5 deg
Target azimuth:  112.4 deg
```

(continues on next page)

(continued from previous page)

```
Light curve name: Tivoli lc
Initial time: 2017-06-22 21:16:00.094 UTC
End time:     2017-06-22 21:28:00.018 UTC
Duration:     11.999 minutes
Time offset:  -0.150 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:15.478 UTC +/- 0.011 seconds
Emersion time:  2017-06-22 21:21:19.838 UTC +/- 0.038 seconds


--------------------------------------------------------------------------------
Site: Windhoek
Geodetic coordinates: Lon: 17d06m31.9s, Lat: -22d41m55.16s, height: 1.902 km
Target altitude: 57.4 deg
Target azimuth:  113.4 deg

Light curve name: Windhoek C14 lc
Initial time: 2017-06-22 21:12:48.250 UTC
End time:     2017-06-22 21:32:47.963 UTC
Duration:     19.995 minutes
Time offset:  -0.375 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:17.234 UTC +/- 0.024 seconds
Emersion time:  2017-06-22 21:21:27.189 UTC +/- 0.026 seconds


--------------------------------------------------------------------------------
Site: Windhoek
Geodetic coordinates: Lon: 17d06m31.9s, Lat: -22d41m55.16s, height: 1.902 km
Target altitude: 57.4 deg
Target azimuth:  113.4 deg
```

(continues on next page)

(continued from previous page)

```
Light curve name: Windhoek D16 lc
Initial time: 2017-06-22 21:20:01.884 UTC
End time:     2017-06-22 21:22:21.894 UTC
Duration:     2.333 minutes
Time offset:  0.000 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:17.288 UTC +/- 0.028 seconds
Emersion time:  2017-06-22 21:21:27.228 UTC +/- 0.034 seconds


################################################################################
                              NEGATIVE OBSERVATIONS
################################################################################

--------------------------------------------------------------------------------
Site: Hakos
Geodetic coordinates: Lon: 16d21m41.32s, Lat: -23d14m11.04s, height: 1.843 km
Target altitude: 56.8 deg
Target azimuth:  112.5 deg

Light curve name: Hakos lc
Initial time: 2017-06-22 21:10:19.461 UTC
End time:     2017-06-22 21:30:19.345 UTC
Duration:     19.998 minutes
Time offset:  -0.200 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion and emersion times were not fitted or instantiated.


################################################################################
                                    RESULTS
```

(continues on next page)

(continued from previous page)

```
################################################################################

Fitted Ellipse:
equatorial_radius: 138.657 +/- 0.373
center_f: -13.613 +/- 0.120
center_g: -2.094 +/- 0.499
oblateness: 0.086 +/- 0.003
position_angle: 123.956 +/- 1.496
polar_radius: 126.769 km
equivalent_radius: 132.579 km
geometric albedo (V): 0.060 (6.0%)

Minimum chi-square: 12.130
Number of fitted points: 10
Number of fitted parameters: 5
Minimum chi-square per degree of freedom: 2.426
Radial dispersion: -0.232 +/- 0.797 km
Radial error:      0.532 +/- 0.222 km

Ephemeris offset (km): X = -13.6 km +/- 0.1 km; Y = -2.1 km +/- 0.5 km
Ephemeris offset (mas): da_cos_dec = -1.280 +/- 0.011; d_dec = -0.197 +/- 0.047

Astrometric object position at time 2017-06-22 21:18:48.200 for reference 'geocenter'
RA = 18 55 15.6523911 +/- 0.034 mas; DEC = -31 31 21.622094 +/- 0.058 mas
```

You can find more information about each Class at their specific Jupyter-Notebook.

The END

CHAPTER

EIGHT

EXAMPLES

SORA is a Python-based, object-oriented library for optimal analysis of stellar occultation data. The user can use this library to build pipelines to analyse their stellar occultation's data. Here follows some key examples on how SORA can be used.

Warning: This page is under development.

8.1 Plotting occultation maps

To generate prediction or post-fit occultation maps, the function to be used is *sora.prediction.plot_occ_map()*. This function receives many required parameters that states the orientation and path of the occultation. When using this function within sora.occ_fit.plot_occ_map() or sora.Prediction.plot_occ_map() all the required parameters are automatically send to the original function. But the user is still able to pass many kwargs are used to configure the occultation map. Without any configuration parameter, the map will have the plain view of the Earth and path. For example.

```
### from a Prediction Table of Phoebe
occs.plot_occ_map()
## Phoebe_2018-06-19T04:36:56.400.png generated
```

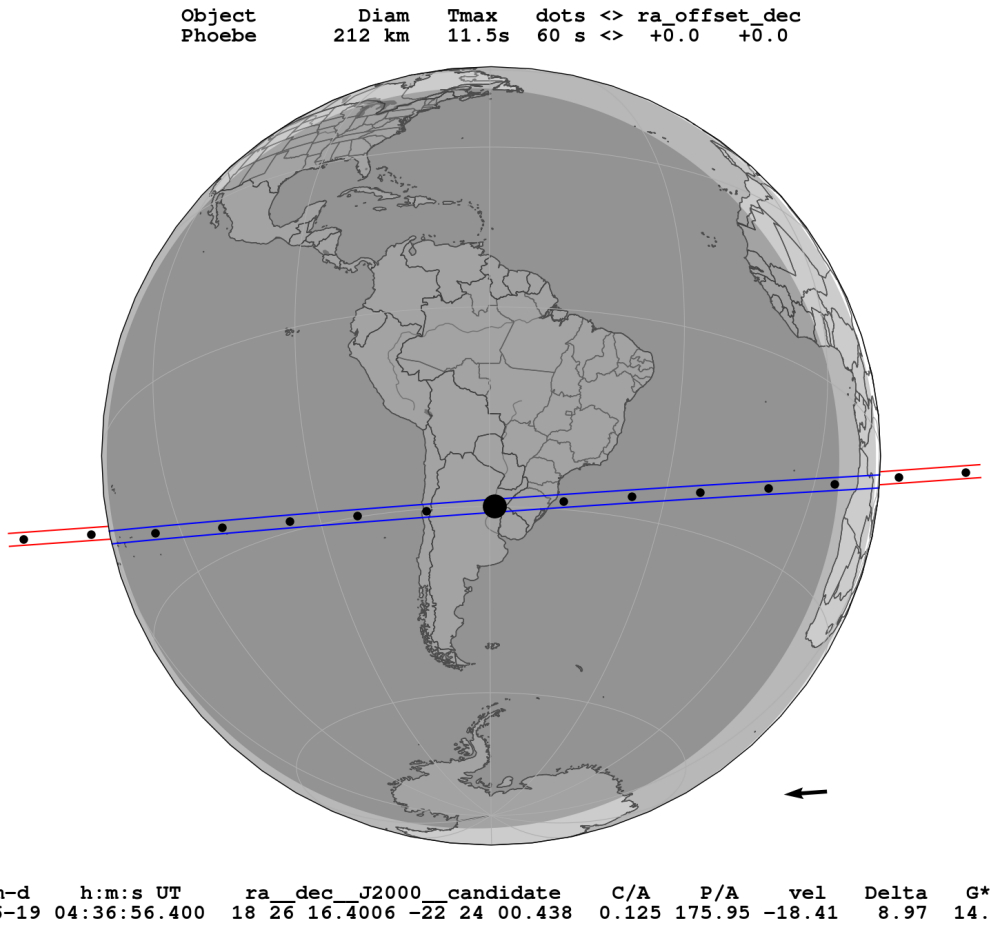


Figure 1: Occultation map of Phoebe, plain view, default configuration.

- nameimg: Change the name of the imaged saved.

```
occs.plot_occ_map(nameimg='Phoebe_teste')
## Phoebe_teste.png generated
```

- resolution: Identify the type of cartopy feature resolution. "1" means a resolution of "10m", "2" a resolution of "50m" and "3" a resolution of "100m". The default is "2". .

```
occs.plot_occ_map(resolution=1)
occs.plot_occ_map(resolution=3)
```

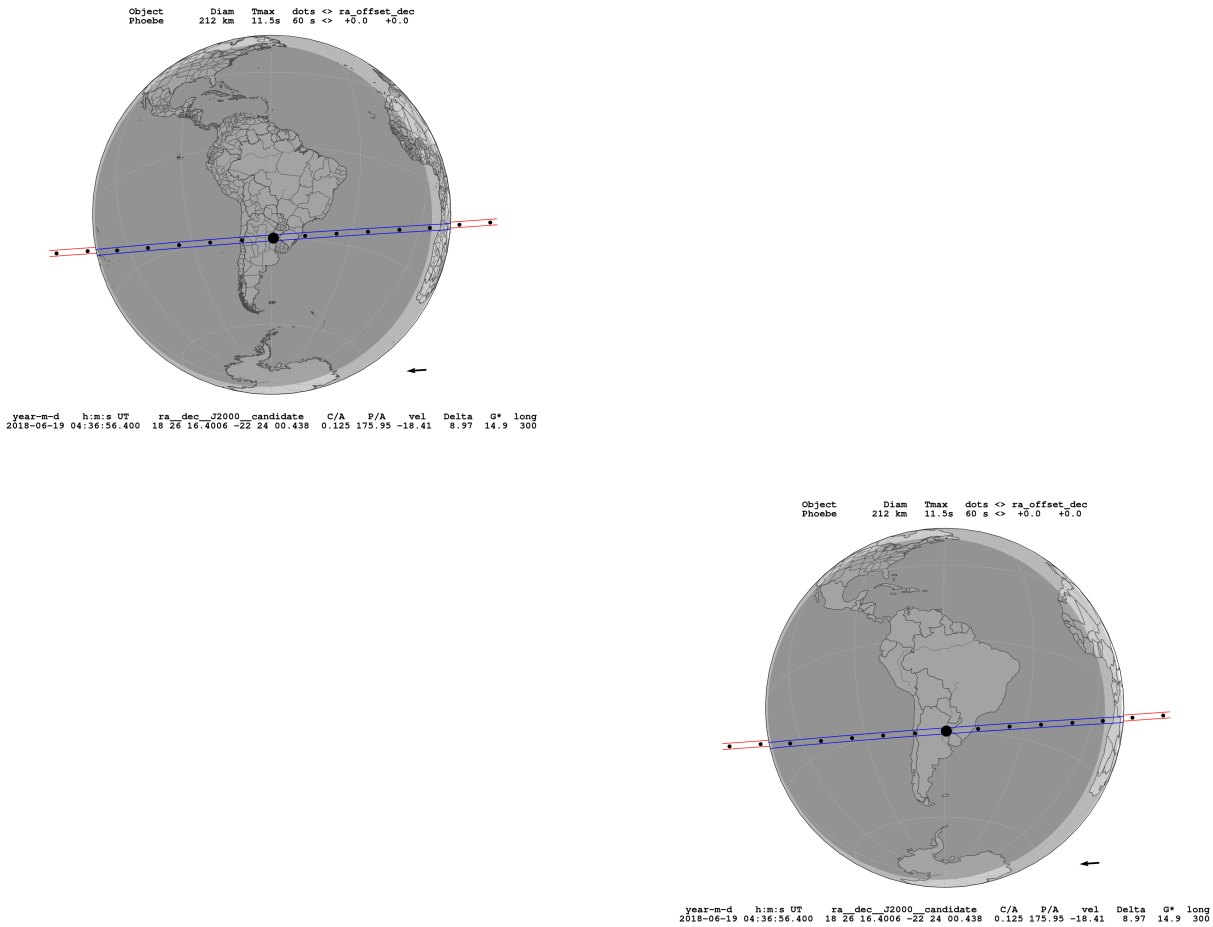


Figure 2: Maps with feature resolution 1 = 10m (left) and 3 = 100m (right).

- states: Plots the state division of the countries. The states of some countries will only be shown depending on the resolution. For instance, USA states are shown for all resolutions, but Brazilian states will be shown only for resolution equal to "1" or "2". This is a cartopy characteristics and cannot be changed. Default=True.

```
occs.plot_occ_map(states=True)
occs.plot_occ_map(states=False)
```

- zoom: Zooms in or out of the map. It must be a number. For the number given in zoom, the dimensions of the map will be divided by that number. For instance, if zoom=2, it will be shown the Earth divided by 2 in X and Y. Default=1. .

```
occs.plot_occ_map(zoom=2)
occs.plot_occ_map(zoom=0.5)
```

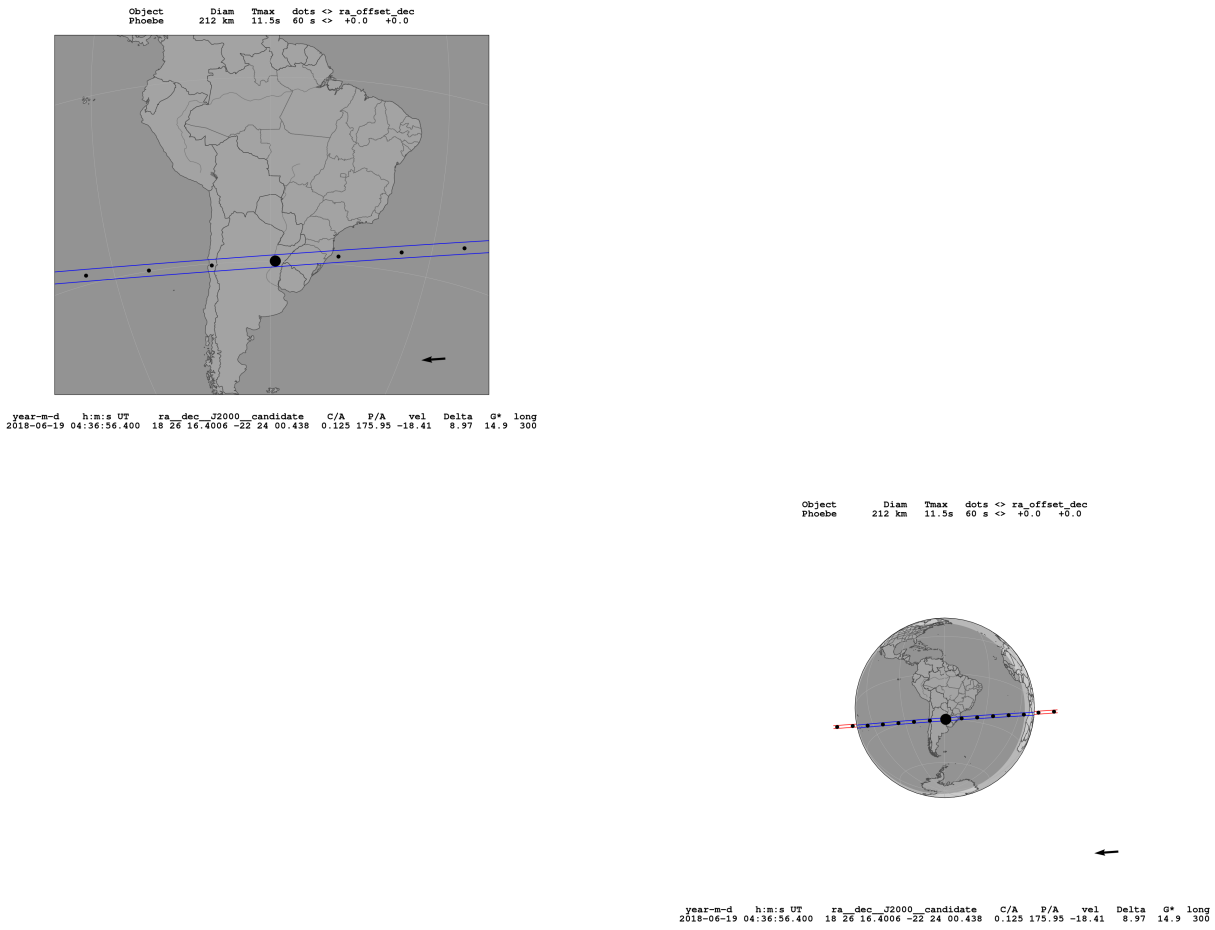


Figure 3: Maps with zoom=2 (left) and zoom=0.5 (right).

- centermap_geo: Center the map given coordinates in longitude and latitude. It must be a list with two numbers. Default=None. If coordinate is on the other side of the map, it gives an error. (left).
- centermap_delta: Displace the center of the map given displacement in X and Y, in km. It must be a list with two numbers. Default=None. (right).

  centermap_geo and centermap_delta are only a displacement on original projection. If Earth rotation is needed, please see centerproj.

```
occs.plot_occ_map(centermap_geo=[0,-40], zoom=2)
# Center the map on longitude=0.0 deg and latitude=-40 deg.
occs.plot_occ_map(centermap_delta=[5000,-1000], zoom=2)
# Displace the center of the map 5000 km East and 1000 km South
```

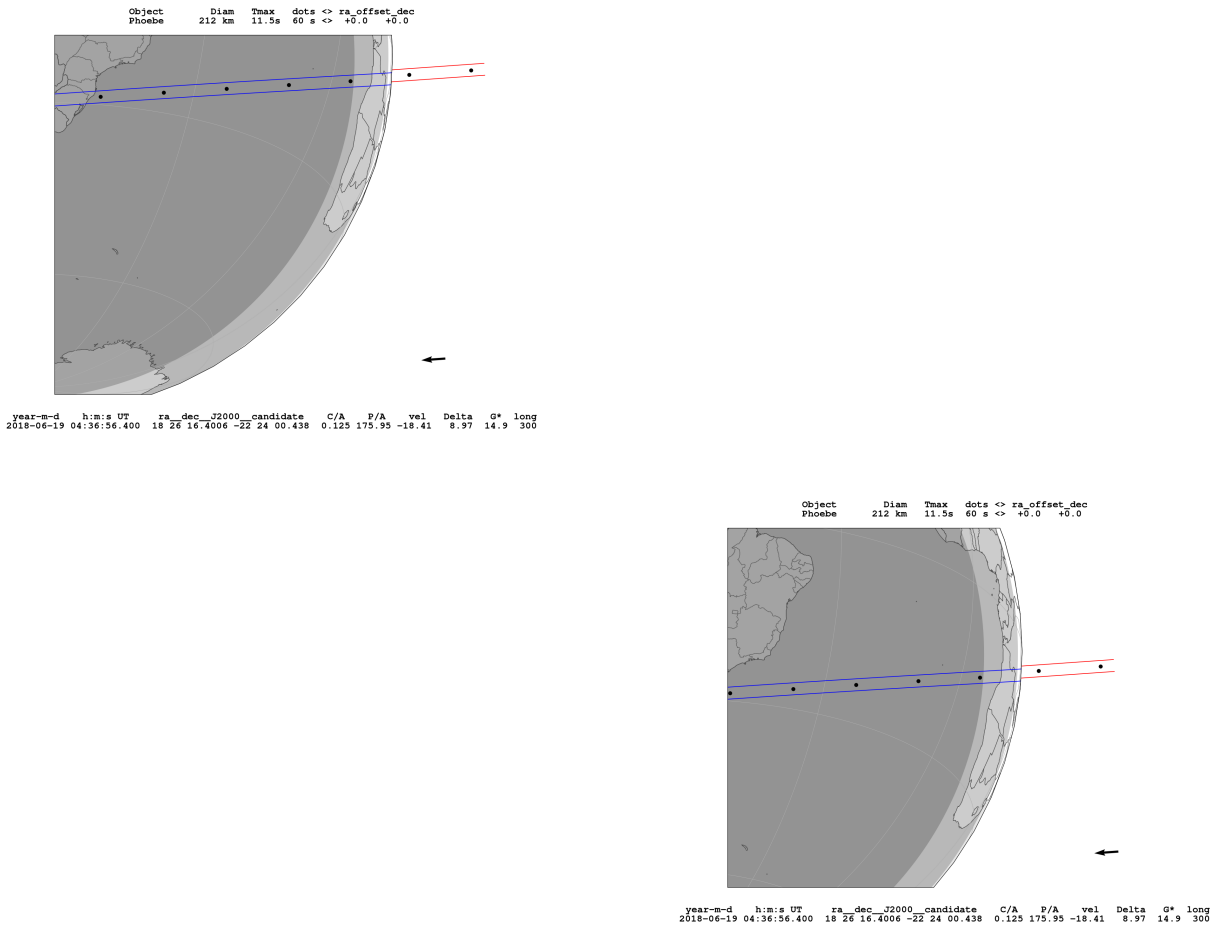


Figure 4: Maps centered on longitude=0.0º and latitude=-40º (left) and with center displaced by 5000 km East and 1000 km South (right)

- centerproj: Rotates the Earth to show occultation with the center projected at a given longitude and latitude. (left).
- labels: Plots text above and below the map with the occultation parameters. Default=True. (right).
- meridians and parallels: Plots lines representing the meridians and parallels given such interval. Default=30 for both parameters. So it will plot lines representing these values each 30º. (right).

```
occs.plot_occ_map(centerproj=[0,-40])
# Rotate to center the map projection on longitude=0.0 deg and latitude=-40 deg.
occs.plot_occ_map(labels=False, meridian=10, parallels=10, zoom=2)
# Displace the center of the map 5000 km East and 1000 km South
```

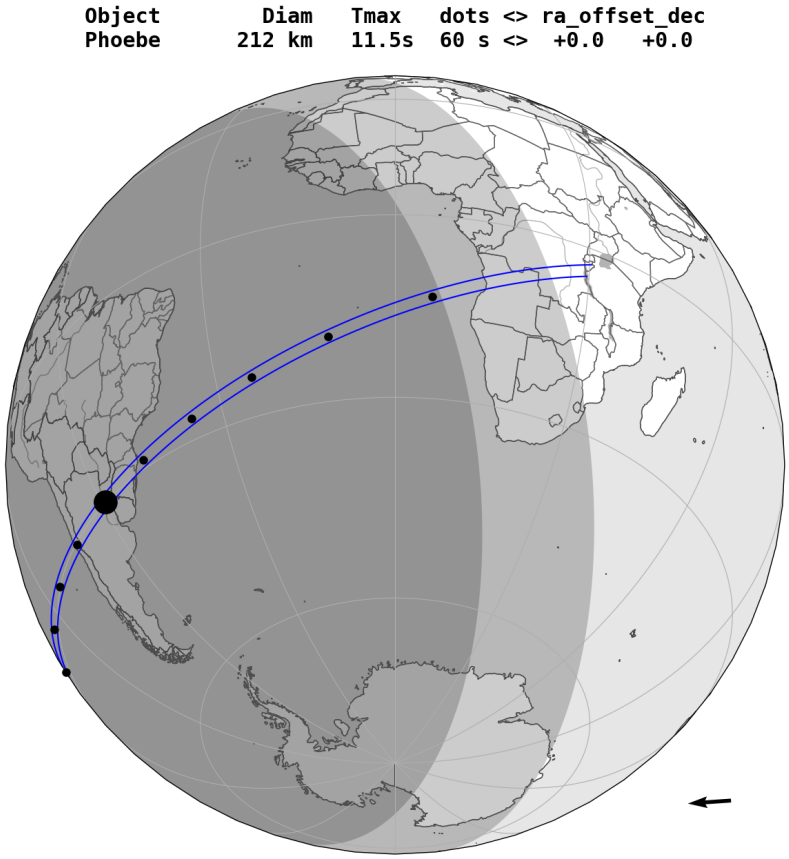


Figure 5: Map with rotated projection to longitude=0.0º and latitude=-40º.

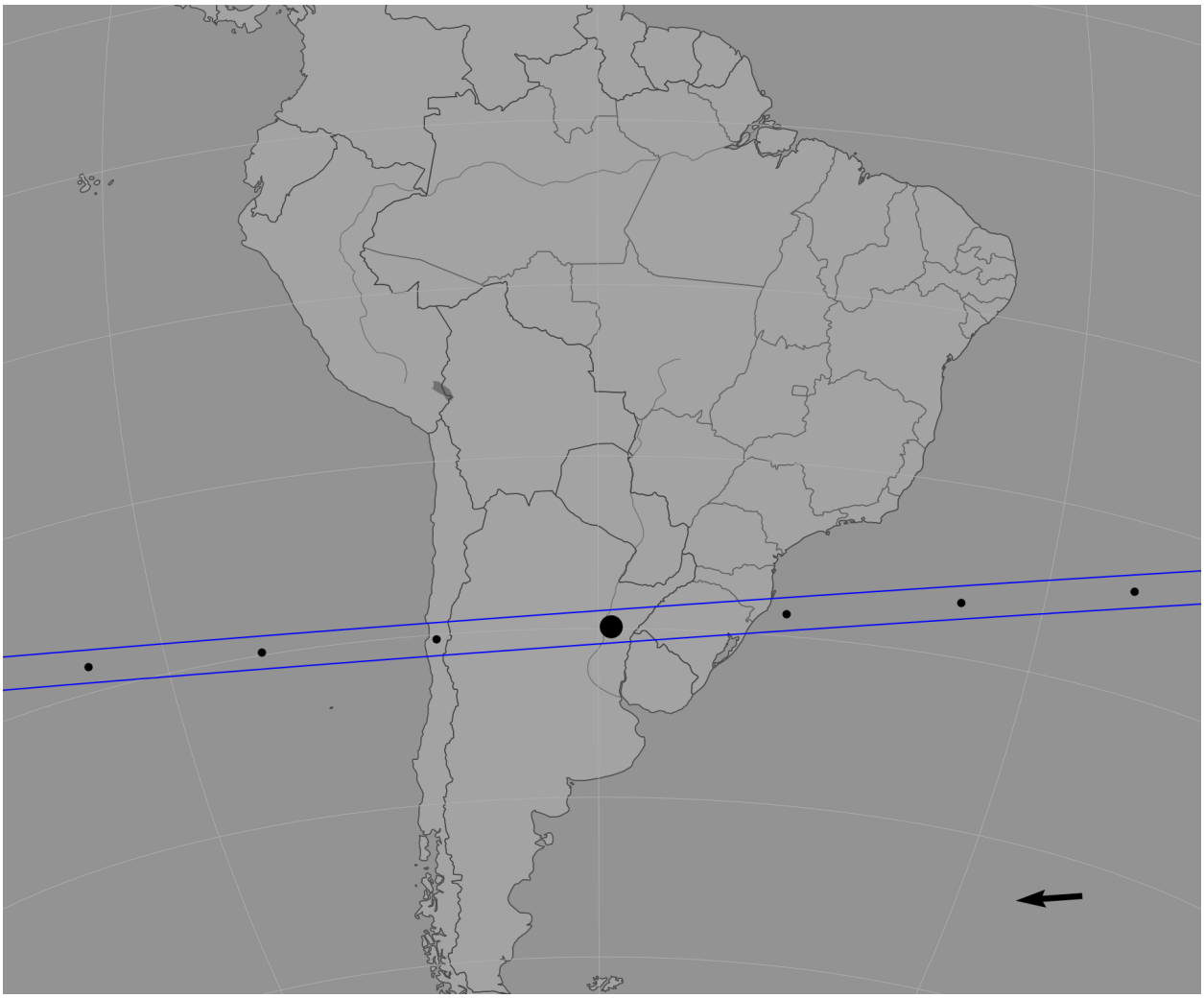


Figure 6: Map without labels, with meridians and parallels each 10º.

- sites: Plots site positions in map. It must be a python dictionary where the key is the name of the site, and the value is a list with longitude, latitude, delta_x, delta_y and color. *delta_x* and *delta_y* are displacement, in km, from the point of the site in the map and the name. *color* is the color of the point. (left).
- countries: Plots the names of countries. It must be a python dictionary where the key is the name of the country and the value is a list with longitude and latitude of the lower left part of the text. (right).

```
sites = {}
sites['Foz'] = [ -54.5936, -25.4347, 10, 10, 'blue']
sites['SOAR'] = [ -70.73919, -30.238027, 10,10,'green']
sites['La Silla'] = [-70.7393888, -29.254611, 10,10,'blue']
occs.plot_occ_map(zoom=5, labels=False, sites=sites)

countries = {}
countries['Brazil'] = [-52.5983973, -23.5570511]
countries['Argentina'] = [-67.2088692, -35.1237852]
occs.plot_occ_map(zoom=3, labels=False, countries=countries, states=False)
```

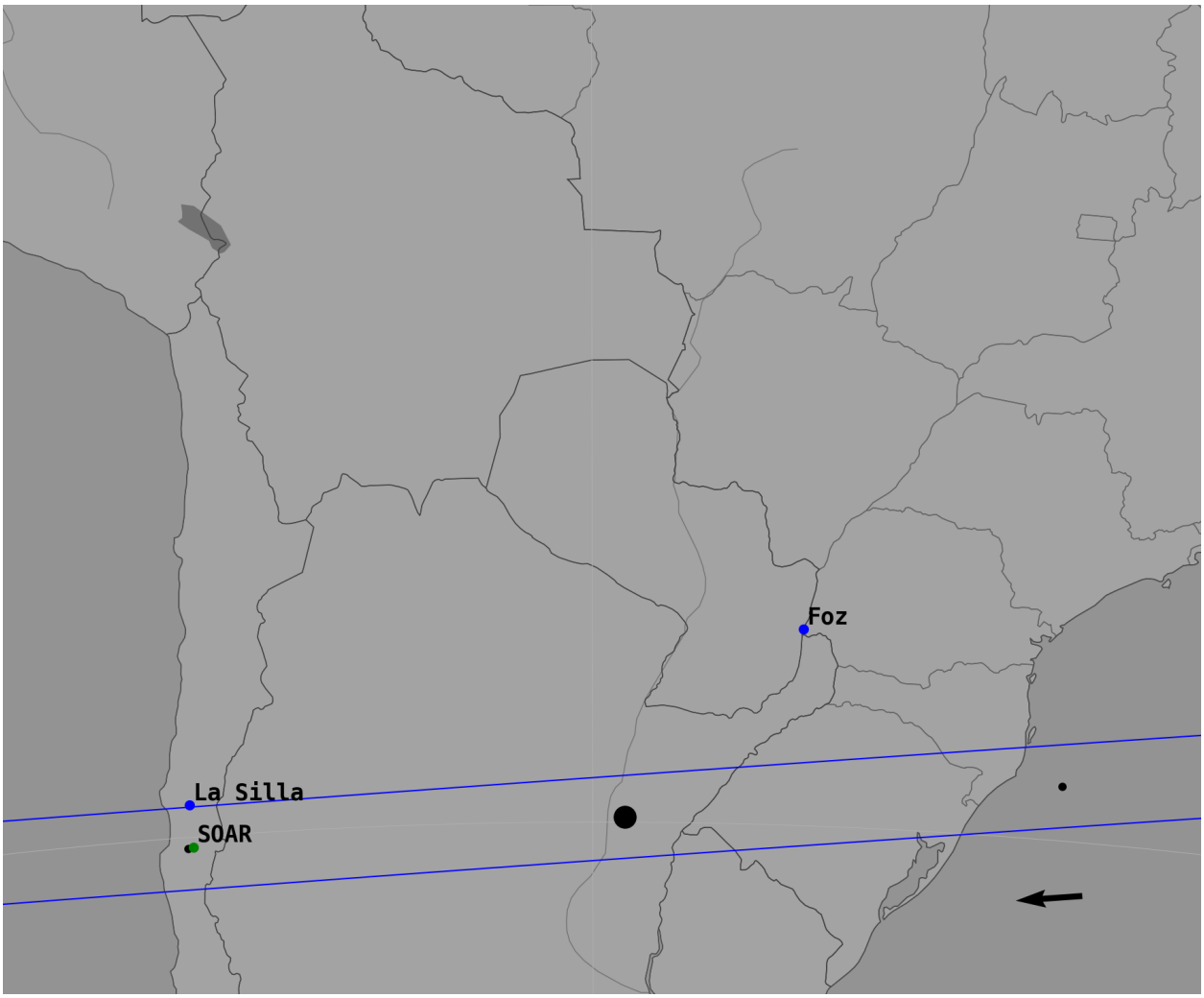


Figure 7: Map with the location of sites.

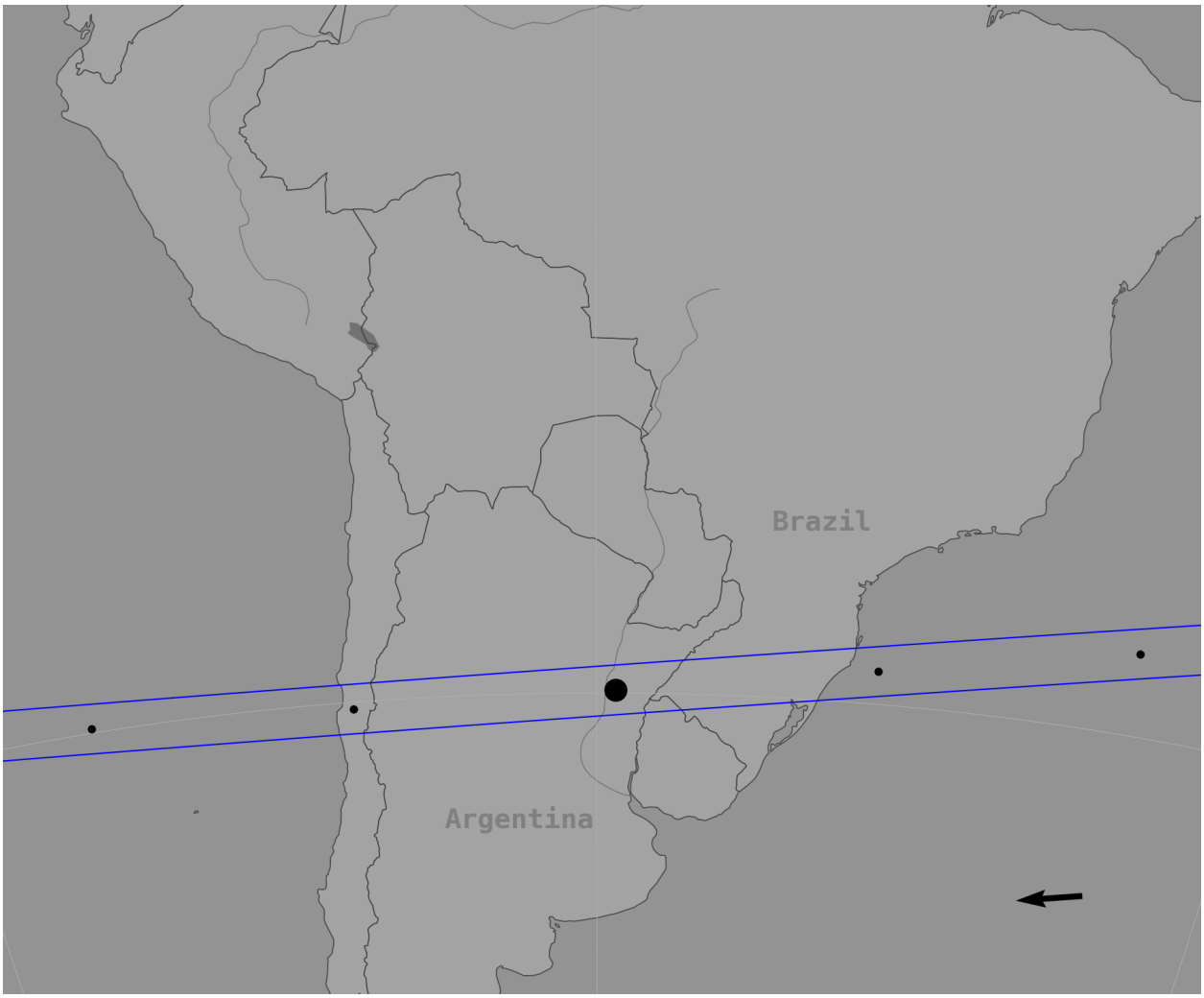


Figure 8: Map with the name of countries (right)

- offset: applies an offset to the ephemeris, calculating new CA and instant of CA. It is a pair of delta_RA*cosDEC and delta_DEC. (left).
- mapstyle: Define the color style of the map. 1 is the default black and white scale. 2 is a colored map. (right).

```
occs.plot_occ_map(zoom=3, offset=[-40,50])
# Applies offsets of -40 mas in Delta_alpha_cos_delta and 50 mas in Delta_delta
occs.plot_occ_map(zoom=3, mapstyle=2)
# Plots a colored map, without offset
```

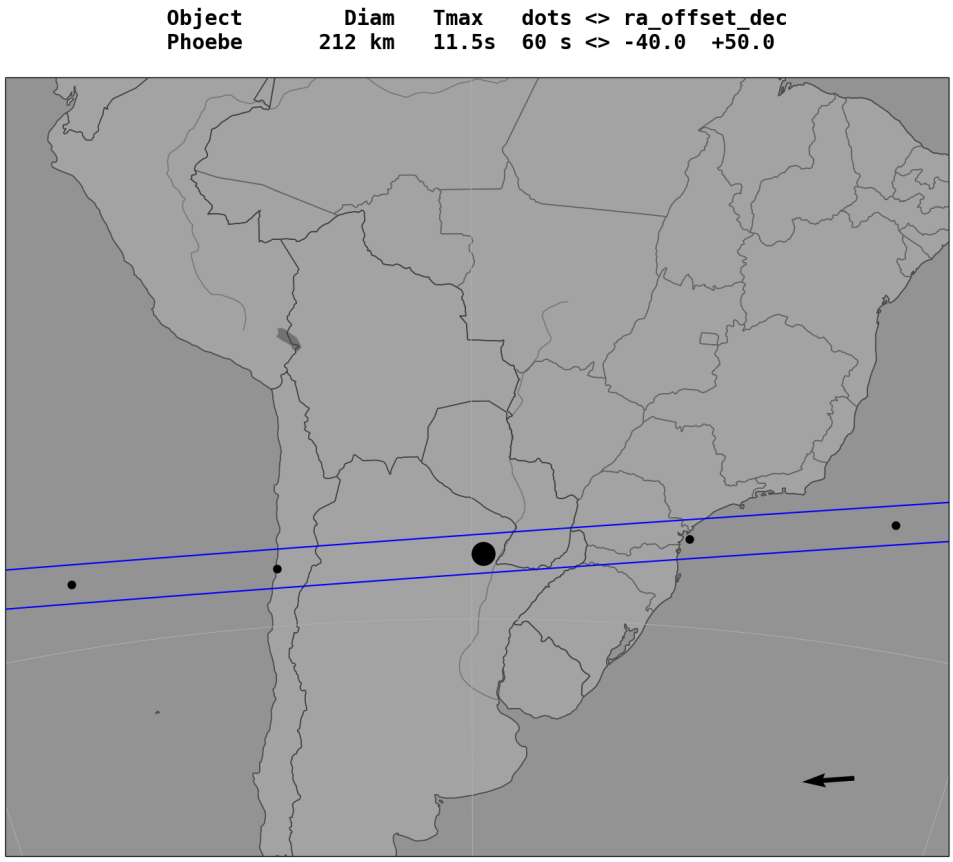


Figure 9: Map with an offset applied.

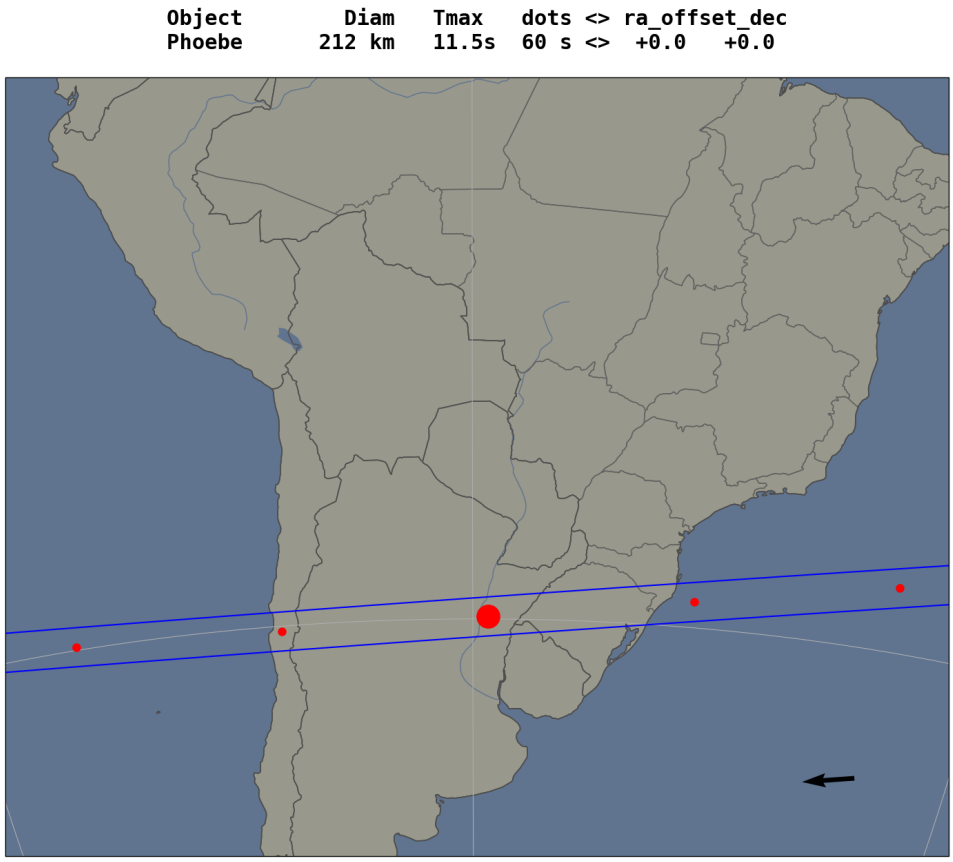


Figure 10: Map with colours.

- error: Ephemeris error in mas. It plots a dashed line representing radius + error. To change the color of these lines, the name of the color must be given to lncolor. (left).
- ring: Similarly to error, it plots a dashed line representing the location of a ring. It is given in km, from the center. To change the color of these lines, the name of the color must be given to rncolor.
- atm: Similarly to error, it plots a dashed line representing the limit of an atmosphere. It is given in km, from the center. To change the color of these lines, the name of the color must be given to atmcolor.
- heights: It plots a circular dashed line showing the locations where the observer would observe the occultation at a given height above the horizons. This must be a list. To change the color of these lines, the name of the color must be given to hcolor. (right).

```
occs.plot_occ_map(zoom=3, labels=False, error=15)
# Shows an error bar of 15 mas
occs.plot_occ_map(heights=[30])
# Shows where the observer will see the occultation with a 30deg height above the␣
↪horizons.
```

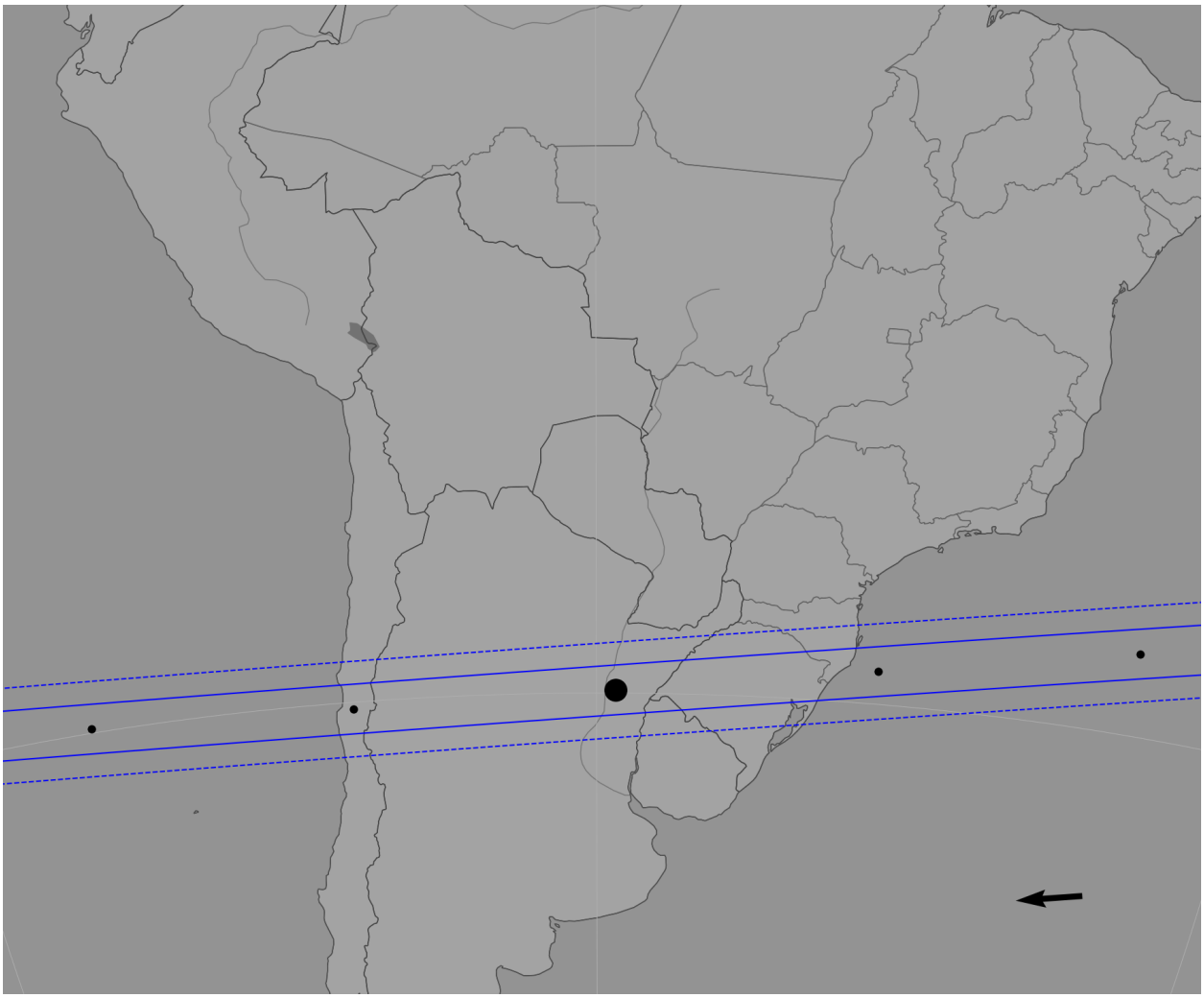


Figure 11: Map showing the error bar.

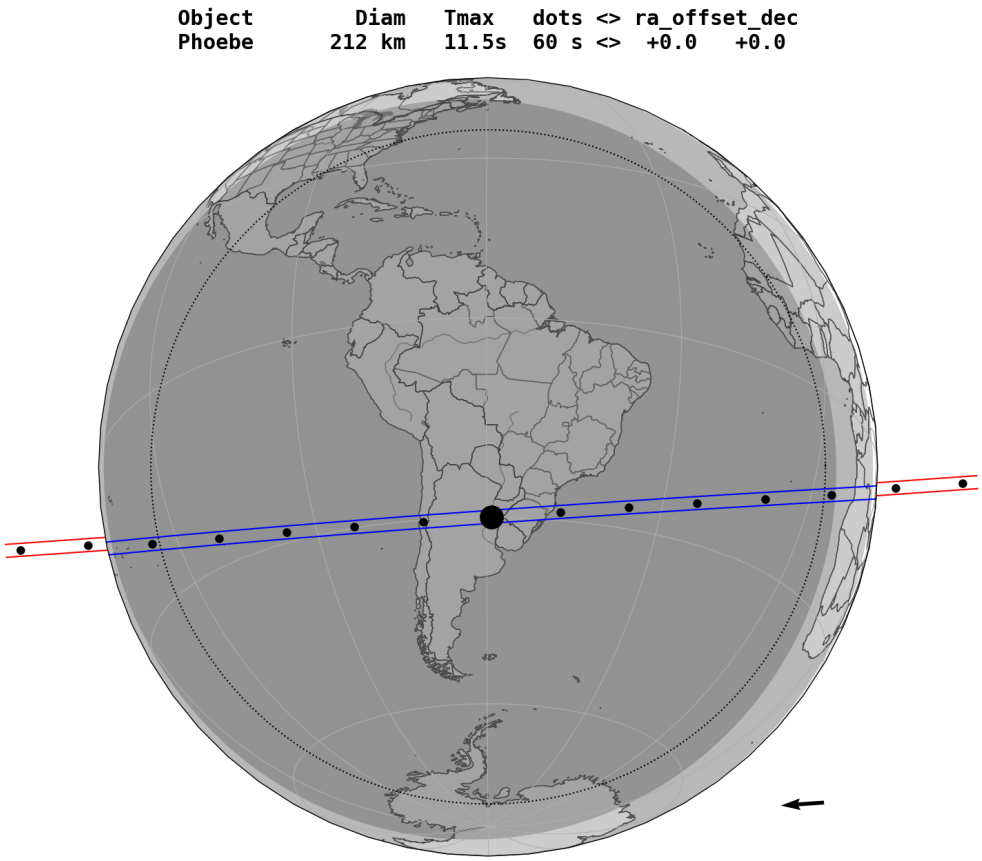


Figure 12: Map with the location that would observe the event with 30º above the horizons.

- mapsize: The size of figure, in cm. It must be a list with two values. Default = [46.0, 38.0].
- cpoints: Interval for the small points marking the center of shadow, in seconds. Default=60. To change the color of these points, the name of the color must be given to ptcolor.
- alpha: The transparency of the night shade, where 0.0 is full transparency and 1.0 is full black. Default = 0.2.
- fmt: The format to save the image. It is parsed directly by matplotlib.pyplot. Default = 'png'.
- dpi: "Dots per inch". It defines the quality of the image. Default = 100.
- nscale, cscale, sscale and pscale: Arbitrary scale for the size for the name of the site, for the name of the country, for the size of point of the site, and the size of the points that represent the center of the shadow, respectively. This scale is arbitrary and is proportional to the size of the image.
- lncolor, outcolor: To change the color of the line that represents the limits of the shadow over Earth and the color of the lines that represents the limits of the shadow outside Earth, respectively.

8.2 Predicting occultations for a spacecraft

The default call of the `prediction()` function will predict stellar occultations for ground-based observers where the occultation parameters calculated are referred to the geocenter. Predicting for a specific observer or a spacecraft is as simple as passing the observer in the parameter `reference_center` of the prediction function.

We must be careful, though. When predicting for the `geocenter` we add the radius of the Earth, the radius of the body and the uncertainty of the body's ephemeris as the radius of search. For a specific observer, the radius of the Earth is not included. Because of this, it might be necessary to give a larger radius and/or a smaller time step for the prediction function. This is important because the coordinates of the ephemeris and star are directly compared in the first iteration for faster identification of potential events, then the occultation parameters are calculated precisely.

In this example, we will predict stellar occultations by Chariklo that could have been observed by the New Horizons in the year 2013, during its transit between Jupiter and Pluto.

[1]:
```
from sora import Body, Spacecraft
from sora.prediction import prediction
```

```
SORA version: 0.2
```

[2]:
```
# Defining the occulting body

chariklo = Body(name='Chariklo', ephem=['10199_Chariklo_nima_v19.bsp', 'de438.bsp'])
print(chariklo)
```

```
Obtaining data for Chariklo from SBDB
################################################################################
                          10199 Chariklo (1997 CU26)
################################################################################
Object Orbital Class: Centaur
Spectral Type:
    SMASS: D  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Relatively featureless spectrum with very steep red slope.
Discovered 1997-Feb-15 by Spacewatch at Kitt Peak

Physical parameters:
Diameter:
    302 +/- 30 km
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),
Rotation:
    7.004 +/- 0 h
    Reference: LCDB (Rev. 2020-October); Warner et al., 2009, [Result based on less than␣
↪full coverage, so that the period may be wrong by 30 percent or so.]  REFERENCE LIST:
↪[Fornasier, S.; Lazzaro, D.; Alvarez-Candal, A.; Snodgrass, C.; et   al. (2014) Astron.
↪ Astrophys. 568, L11.], [Leiva, R.; Sicardy, B.; Camargo, J.I.B.; Desmars, J.; et al.␣
↪(2017)   Astron. J. 154, A159.]
Absolute Magnitude:
    6.7 +/- 0 mag
    Reference: MPO452314,
Albedo:
    0.045 +/- 0.01
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),

----------- Ephemeris -----------
```

(continues on next page)

(continued from previous page)

```
EphemKernel: 10199_CHARIKLO_NIMA_V19/DE438 (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

[3]:
```
# Defining the New Horizons as an Observer.
# The kernel has the path of the spacecraft from 2012-12-31T23:58:52.816 up to 2014-12-
↪08T05:12:10.384
# Another option is to use the JPL/Horizons web service making ephem='horizons'

nh = Spacecraft(name='New Horizons', spkid='-98', ephem=['nh_recon_od117_v01.bsp'])
```

[4]:
```
# Prediction of stellar occultations by Chariklo for the year 2013 as observed by the␣
↪New Horizons.
# The time step of the ephemeris is 10s and the radius of search is 600 km for the quick␣
↪search.
# The full period of search was split in 12 division to avoid overflow of memory caused␣
↪by the ephemeris
# and to avoid download stars for a large region of sky.

pred = prediction(body=chariklo, time_beg='2013-01-01 00:00', time_end='2014-01-01 00:00
↪',
                  radius=600, step=10, divs=12, reference_center=nh)
```

```
Ephemeris was split in 12 parts for better search of stars

Searching occultations in part 1/12
Generating Ephemeris between 2013-01-01 00:00:00.000 and 2013-01-31 09:59:50.000 ...
Downloading stars ...
    1741 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 2/12
Generating Ephemeris between 2013-01-31 10:00:00.000 and 2013-03-02 19:59:50.000 ...
Downloading stars ...
    1749 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 3/12
Generating Ephemeris between 2013-03-02 20:00:00.000 and 2013-04-02 05:59:50.000 ...
Downloading stars ...
    1701 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 4/12
Generating Ephemeris between 2013-04-02 06:00:00.000 and 2013-05-02 15:59:50.000 ...
Downloading stars ...
    1647 Gaia-EDR3 stars downloaded
Identifying occultations ...
```

(continues on next page)

(continued from previous page)

```
Searching occultations in part 5/12
Generating Ephemeris between 2013-05-02 16:00:00.000 and 2013-06-02 01:59:50.000 ...
Downloading stars ...
    1733 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 6/12
Generating Ephemeris between 2013-06-02 02:00:00.000 and 2013-07-02 11:59:50.000 ...
Downloading stars ...
    1636 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 7/12
Generating Ephemeris between 2013-07-02 12:00:00.000 and 2013-08-01 21:59:50.000 ...
Downloading stars ...
    1741 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 8/12
Generating Ephemeris between 2013-08-01 22:00:00.000 and 2013-09-01 07:59:50.000 ...
Downloading stars ...
    1781 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 9/12
Generating Ephemeris between 2013-09-01 08:00:00.000 and 2013-10-01 17:59:50.000 ...
Downloading stars ...
    1908 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 10/12
Generating Ephemeris between 2013-10-01 18:00:00.000 and 2013-11-01 03:59:50.000 ...
Downloading stars ...
    1718 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 11/12
Generating Ephemeris between 2013-11-01 04:00:00.000 and 2013-12-01 13:59:50.000 ...
Downloading stars ...
    1749 Gaia-EDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 12/12
Generating Ephemeris between 2013-12-01 14:00:00.000 and 2013-12-31 23:59:50.000 ...
Downloading stars ...
    1798 Gaia-EDR3 stars downloaded
Identifying occultations ...

3 occultations found.
```

[5]:
```
# printing prediction table
```

(continues on next page)

(continued from previous page)

```
pred.pprint_all()
```

```
         Epoch              ICRS Star Coord at Epoch      Geocentric Object Position    C/
↪A    P/A    Vel     Dist    G      G*   long  loct M-G-T S-G-T Gaia-EDR3 Source ID
                                                                                        ␣
↪arcsec  deg   km / s    AU    mag    mag   deg  hh:mm  deg   deg
---------------------- ---------------------------- ---------------------------- ---
↪--- ------ ------ ------- ------ ------ ---- ----- ----- ----- --------------------
2013-05-08 08:56:11.580 08 14 06.33337 +00 44 18.72873 08 14 06.33419 +00 44 18.74317  0.
↪019  40.42  -8.01  15.717 18.527 17.533  123 17:08    97    79 3089568957554344960
2013-07-13 13:37:48.260 08 10 44.43129 +01 27 22.86678 08 10 44.43275 +01 27 22.89235  0.
↪034  40.52  -7.99  15.978 17.950 16.954  347 12:44    47    22 3089856131952198272
2013-10-12 00:03:14.560 08 06 15.00058 +02 25 00.19559 08 06 14.99944 +02 25 00.17579  0.
↪026 220.61  -7.96  16.347 15.812 14.811  100 06:43   161    76 3090959839765902336
```

The column `Geocentric Object Position` gives the object position for the observer, not the geocenter. It is maintained like this for backward compatibility, but it will be fixed in SORAv1.0.

There were 3 stellar occultations by Chariklo that could have been observed by New Horizons in 2013.

The plots of maps in SORA is still projected for an occultation on Earth. Plots for a spacecraft may be implemented in the future.

8.3 Using different Vizier Catalogues in prediction() and Star()

As the current most precise catalog available, Gaia is the default catalog implemented in SORA. Thus, the user can make predictions or adding a star in the occultation procedure by providing the coordinates or the Gaia source ID of the targets. The user can also choose between Gaia-DR2 and Gaia-EDR3 as the source catalog. However, Gaia does not have the brightest stars on the sky in its list, which can be found in Hipparcos, for instance. In other cases, some of the faintest stars are also not available, and may be available in other catalogs.

Regarding this problem, from SORA v0.3 onwards can allow the user to implement any new catalog available on Vizier that can be used in the prediciton() function and the Star() class.

To add a new catalog, the user has to provide the keywords for the necessary parameters allowing SORA to retrieve the correct values.

[1]:
```
from sora.star.catalog import VizierCatalogue
from astropy.time import Time
from sora import Star
```

```
SORA version: 0.3
```

[2]:
```
# Defining the NOMAD catalogue from Vizier
nomad = VizierCatalogue(name='NOMAD', cat_path='I/297/out', code='NOMAD1', ra='RAJ2000',␣
↪dec='DEJ2000',
                        epoch=Time('J2000'), band={'B': 'Bmag', 'V': 'Vmag', 'R': 'Rmag',
↪ 'J': 'Jmag', 'H': 'Hmag', 'K': 'Kmag'})
```

Note the parameters `name` (name of the catalogue) and `cat_path` (path of the catalogue on Vizier) are required parameters for the proper identification of the catalog. The parameters `code`, `ra`, `dec` and `epoch` are required for a proper identification of the star. The `code` parameter is a unique identifier within the catalog. In Gaia, for instanca, `code="Source"`.

The parameters for the other astrometric parameters can also be given, like `pmra` and `pmdec` for the proper motions PM_RA*COS(DEC) and PM_DEC, `parallax` and `rad_vel` for parallax and radial velocity. If not given, these parameters are default to zero. Finally, if the user wants to characterize the stars' magnitudes, the `band` parameter can be given by a dictionary with the keywords for all the bands available, as shown for the NOMAD catalog above.

8.3.1 Predicting with new catalog

To predict with a different catalogue than Gaia, just pass the new created VizierCatalogue object to the parameter `catalogue` in the prediction() function

[3]:
```
from sora import Body
from sora.prediction import prediction

mars = Body(name='Mars', orbit_class='planet', diameter=6779, ephem='horizons', spkid=
↪'499', database=None)

p = prediction(body=mars, time_beg='2022-09-10 00:00', time_end='2022-09-10 06:00',␣
↪catalogue=nomad)
p.pprint_all()
```

```
Ephemeris was split in 1 parts for better search of stars

Searching occultations in part 1/1
Generating Ephemeris between 2022-09-10 00:00:00.000 and 2022-09-10 05:59:00.000 ...
Downloading stars ...
    54 NOMAD stars downloaded
Identifying occultations ...

12 occultations found.
         Epoch             ICRS Star Coord at Epoch      Geocentric Object Position    C/
↪A    P/A    Vel     Dist    B      V      R      J      H      K    long  loct M-G-T S-
↪G-T NOMAD Source ID

                                                                                  ␣
↪arcsec  deg   km / s    AU    mag    mag    mag    mag    mag    mag   deg  hh:mm  deg␣
↪  deg
----------------------- ------------------------------ ------------------------------ ---
↪--- ------ ------ ------- ------ ------ ------ ------ ------ ------ ---- ----- ----- --
↪--- ---------------
2022-09-10 00:37:10.480 04 37 55.92334 +20 56 19.88016 04 37 55.84499 +20 56 25.91258  6.
↪131 349.69  13.46   0.906 20.750    nan 19.880    nan    nan    nan   71 05:21    89  ␣
↪ 96    1109-0056128
2022-09-10 00:39:55.360 04 37 56.13199 +20 56 22.67016 04 37 56.08202 +20 56 26.51681  3.
↪910 349.69  13.46   0.906    nan    nan 20.270    nan    nan    nan   70 05:21    89  ␣
↪ 96    1109-0056132
2022-09-10 00:45:26.420 04 37 56.38267 +20 56 41.21988 04 37 56.55793 +20 56 27.72996 13.
↪712 169.68  13.46   0.906 17.910 17.060 17.150 14.587 14.015 13.720   69 05:21    89  ␣
↪ 96    1109-0056134
2022-09-10 01:04:25.900 04 37 58.33666 +20 56 21.05988 04 37 58.19585 +20 56 31.90470 11.
↪023 349.69  13.45   0.906 21.280    nan 19.820    nan    nan    nan   64 05:21    89  ␣
↪ 96    1109-0056139
2022-09-10 01:29:09.520 04 38 00.44400 +20 56 28.41000 04 38 00.32811 +20 56 37.33836  9.
↪075 349.69  13.45   0.906 21.050    nan 19.580    nan    nan    nan   58 05:21    89  ␣
↪ 96    1109-0056148
```

(continues on next page)

(continued from previous page)

```
2022-09-10 01:34:37.760 04 38 00.78334 +20 56 39.81012 04 38 00.79981 +20 56 38.54023  1.
↪291 169.70  13.45   0.906 19.650 17.970 18.610 16.472 16.437 15.481   57 05:21    88  ␣
↪ 96    1109-0056152
2022-09-10 02:20:09.040 04 38 04.59353 +20 56 58.59492 04 38 04.72410 +20 56 48.53684 10.
↪223 169.69  13.44   0.906    nan    nan    nan 17.123 16.219 15.762   45 05:21    88  ␣
↪ 96    1109-0056165
2022-09-10 02:37:08.420 04 38 06.11633 +20 56 57.82092 04 38 06.18842 +20 56 52.26594  5.
↪646 169.70  13.44   0.906 16.350 15.490 16.100 13.865 13.381 13.213   41 05:21    88  ␣
↪ 96    1109-0056170
2022-09-10 03:16:45.980 04 38 09.66934 +20 56 55.84992 04 38 09.60305 +20 57 00.95958  5.
↪193 349.70  13.43   0.906 18.350 17.830 18.500 16.449 16.481 15.707   31 05:21    88  ␣
↪ 96    1109-0056189
2022-09-10 03:32:17.940 04 38 10.97573 +20 57 01.70892 04 38 10.94127 +20 57 04.36582  2.
↪700 349.70  13.43   0.906 14.070 13.250 13.480 10.624  9.952  9.793   27 05:21    87  ␣
↪ 96    1109-0056199
2022-09-10 04:08:19.700 04 38 14.14934 +20 57 04.20012 04 38 14.04481 +20 57 12.26359  8.
↪195 349.71  13.43   0.906 20.230    nan 19.970    nan    nan    nan   18 05:21    87  ␣
↪ 96    1109-0056210
2022-09-10 05:58:45.300 04 38 23.44133 +20 57 44.97984 04 38 23.55198 +20 57 36.44106  8.
↪678 169.71  13.40   0.905 18.350 17.620 18.720 16.246 15.967 15.718  351 05:21    86  ␣
↪ 96    1109-0056239
```

If `catalogue` is not given, then the default catalogue is the Gaia-EDR3. The `catalogue` parameter can also be "gaiadr2" or "gaiaedr3" as it has been the case since SORA v0.2.

Note that all the bands given to the nomad object were downloaded and included in the PredictionTable. Those stars where a given band is not defined is shown with "Not A Number" (nan) mark.

In order to restrict the prediction function based on a magnitude limit, the user has to define the `mag_lim` parameter in the prediction() function. Previously to SORA v0.3, only a number could be given to the parameter. Currently, if a number is passed, the limit of magnitude will be defined only for the first band defined. In this example, it is the "B" band. To define which bands are to be limited, the user must pass a dictionary to the `mag_lim` parameter. For instance, the next piece of code predicts occultations limiting stars with the V band smaller than 16 and the H band smaller than 14.

[4]:
```
p = prediction(body=mars, time_beg='2022-09-10', time_end='2022-09-11', catalogue=nomad,␣
↪mag_lim={'V': 16, 'H': 14})
p.pprint_all()
```

```
Ephemeris was split in 1 parts for better search of stars

Searching occultations in part 1/1
Generating Ephemeris between 2022-09-10 00:00:00.000 and 2022-09-10 23:59:00.000 ...
Downloading stars ...
    39 NOMAD stars downloaded
Identifying occultations ...

4 occultations found.
         Epoch             ICRS Star Coord at Epoch      Geocentric Object Position    C/
↪A    P/A    Vel     Dist    B      V      R      J      H      K     long  loct M-G-T S-
↪G-T NOMAD Source ID

                                                                                      ␣
↪arcsec  deg   km / s    AU    mag    mag    mag    mag    mag    mag    deg  hh:mm  deg␣
↪  deg
```

(continues on next page)

(continued from previous page)

```
---------------------- ------------------------------ ------------------------------ ---
↪--- ------ ------ ------- ------ ------ ------ ------ ------ ------ ---- ----- ----- --
↪--- ---------------
2022-09-10 02:37:08.420 04 38 06.11633 +20 56 57.82092 04 38 06.18842 +20 56 52.26594  5.
↪646 169.70  13.44   0.906 16.350 15.490 16.100 13.865 13.381 13.213   41 05:21    88  ␣
↪ 96    1109-0056170
2022-09-10 03:32:17.940 04 38 10.97573 +20 57 01.70892 04 38 10.94127 +20 57 04.36582  2.
↪700 349.70  13.43   0.906 14.070 13.250 13.480 10.624  9.952  9.793   27 05:21    87  ␣
↪ 96    1109-0056199
2022-09-10 12:09:07.580 04 38 55.57073 +20 58 42.82284 04 38 55.38487 +20 58 57.21908 14.
↪630 349.75  13.35   0.904 13.120 12.500 12.970 10.763 10.373 10.215  258 05:20    82  ␣
↪ 96    1109-0056366
2022-09-10 18:23:28.760 04 39 27.62599 +21 00 06.62004 04 39 27.47466 +21 00 18.37673 11.
↪946 349.78  13.28   0.902 16.220 14.990 14.840 11.974 11.311 11.111  164 05:20    79  ␣
↪ 96    1110-0056967
```

8.3.2 Using a not Gaia star in the occultation process

To include a star in the occultation procedure, we must use the Star() class to define a single star. The standard catalogs defined in this class are Gaia-DR2 and Gaia-EDR3. However, we can add a star from a different catalogue defined in Vizier using the VizierCatalogue() class just like we did in the prediction() function. Notice that, in this case, we are not able to use the shortcut of the Occultation() class by providing the direct coordinate of the star. We must define a Star object previously.

[5]:
```
# The Star class already reads the nomad catalogue to download the magnitudes in the␣
↪available bands.
# In the case of this example, we must set it to False, or it will download the bands as␣
↪the main catalogue
# and the auxiliary one, replacing the magnitude with the same values.
s = Star(code='1109-0056199', catalogue=nomad, nomad=False)
print(s)
```

```
1 NOMAD star found band={'B': 14.069999694824219, 'V': 13.25, 'R': 13.479999542236328, 'J
↪': 10.62399959564209, 'H': 9.95199966430664, 'K': 9.793000221252441}
star coordinate at J2000.0: RA=4h38m10.97573s +/- 0.0 mas, DEC=20d57m01.7089s +/- 0.0 mas
NOMAD star Source ID: 1109-0056199
ICRS star coordinate at J2000.0:
RA=4h38m10.97573s +/- 0.0000 mas, DEC=20d57m01.7089s +/- 0.0000 mas
pmRA=0.000 +/- 0.000 mas/yr, pmDEC=0.000 +/- 0.000 mas/yr
Plx=0.0000 +/- 0.0000 mas, Rad. Vel.=0.00 +/- 0.00 km/s

Magnitudes: B: 14.070, V: 13.250, R: 13.480, J: 10.624, H:  9.952, K:  9.793

Apparent diameter from Kervella et. al (2004):
    V: 0.0660 mas, B: 0.0611 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0596 mas, V: 0.0616 mas
    ms: B: 0.0844 mas, V: 0.0579 mas
    vs: B: 0.0848 mas, V: 0.0781 mas
```

Usually, when working with the occultation we want to estimate the uncertainty on the position of the star at the epoch of occultation. For this, the user can provide the list of keywords for the uncertainty of each of the astrometric parameters

(if not available, None must be passed for the parameter uncertainty). An important note is that only with the Gaia catalog SORA will obtain the full covariance matrix.

[6]:
```
nomad = VizierCatalogue(name='NOMAD', cat_path='I/297/out', code='NOMAD1', ra='RAJ2000',␣
↪dec='DEJ2000', pmra='pmRA', pmdec='pmDE',
                        epoch=Time('J2000'), band={'B': 'Bmag', 'V': 'Vmag', 'R': 'Rmag',
↪ 'J': 'Jmag', 'H': 'Hmag', 'K': 'Kmag'},
                        errors=['e_RAJ2000', 'e_DEJ2000', 'e_pmRA', 'e_pmDE', None,␣
↪None])

s = Star(code='1109-0056199', catalogue=nomad, verbose=False, nomad=False)
print(s)
```

```
NOMAD star Source ID: 1109-0056199
ICRS star coordinate at J2000.0:
RA=4h38m10.97573s +/- 31.0000 mas, DEC=20d57m01.7089s +/- 17.0000 mas
pmRA=-7.100 +/- 6.400 mas/yr, pmDEC=-16.600 +/- 6.300 mas/yr
Plx=0.0000 +/- 0.0000 mas, Rad. Vel.=0.00 +/- 0.00 km/s

Magnitudes: B: 14.070, V: 13.250, R: 13.480, J: 10.624, H:  9.952, K:  9.793

Apparent diameter from Kervella et. al (2004):
    V: 0.0660 mas, B: 0.0611 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0596 mas, V: 0.0616 mas
    ms: B: 0.0844 mas, V: 0.0579 mas
    vs: B: 0.0848 mas, V: 0.0781 mas
```

[7]:
```
from sora import Occultation

occ = Occultation(star=s, body=mars, time='2022-09-10 03:32')
print(occ)
```

```
Stellar occultation of star NOMAD 1109-0056199 by Mars.

Geocentric Closest Approach: 3.042 arcsec
Instant of CA: 2022-09-10 03:32:06.900
Position Angle: 349.70 deg
Geocentric shadow velocity: 13.43 km / s
Sun-Geocenter-Target angle:  96.16 deg
Moon-Geocenter-Target angle: 87.36 deg


No observations reported

################################################################################
                                    STAR
################################################################################
NOMAD star Source ID: 1109-0056199
ICRS star coordinate at J2000.0:
RA=4h38m10.97573s +/- 31.0000 mas, DEC=20d57m01.7089s +/- 17.0000 mas
pmRA=-7.100 +/- 6.400 mas/yr, pmDEC=-16.600 +/- 6.300 mas/yr
Plx=0.0000 +/- 0.0000 mas, Rad. Vel.=0.00 +/- 0.00 km/s
```

(continues on next page)

(continued from previous page)

```
Magnitudes: B: 14.070, V: 13.250, R: 13.480, J: 10.624, H:  9.952, K:  9.793

Apparent diameter from Kervella et. al (2004):
    V: 0.0660 mas, B: 0.0611 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0596 mas, V: 0.0616 mas
    ms: B: 0.0844 mas, V: 0.0579 mas
    vs: B: 0.0848 mas, V: 0.0781 mas

Geocentric star coordinate at occultation Epoch (2022-09-10 03:32:06.900):
RA=4h38m10.96423s +/- 149.1636 mas, DEC=20d57m01.3323s +/- 70.4248 mas

################################################################################
                                      Mars
################################################################################
Object Orbital Class: Planet


Physical parameters:
Diameter:
    6779 +/- 0 km
    Reference: User,

----------- Ephemeris -----------

EphemHorizons: Ephemeris are downloaded from Horizons website (SPKID=499)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

8.4 Fitting occultation chord to 3D shape models

Usually, occultation chords are fitted to simples shapes like circles and ellipses. However, for small bodies this may not be good enough, since they are not expected to be rounded bodies. In some cases, we have a 3D shape model defined from rotational lightcurve inversion, and we want to compare numericaly the occultation chords with these models.

For this, since SORA v0.3.0, a method was implemented to include 3D shape models to be fitted to the occultations.

Important note: SORA does not derive a 3D shape model. It uses 3D shape models already derived from other sources to be fitted with occultation chords.

In this example, we will use the 3D shape model of Phoebe and fit the occultation chords observed on July 06th, 2017. The results here presented are based, although not equal to, the occultation published by Gomes-Júnior et al. (2020) and available on A&A and Arxiv

To replicate the process shown below, it is necessary to download the 3D shape model of Phoebe here. The file must be the in "OBJ" format. Either the 50k or 200k versions will work, however the 200k one will demand more processing. The surface texture can also be downloaded and included in the process, but it is not relevant for the fitting process, only for plottings.

Important note: To see the details for all the methods available in the `Shape3D` class, please refer to the body guideline.

[1]:
```
# The following commands download and save the 3D shape model from Gaskell.
import requests
response = requests.get("https://3d-asteroids.space/data/moons/models/S9_Phoebe.obj")
open("Phoebe_Gaskell_50k_poly.obj", "wb").write(response.content)
response = requests.get("https://3d-asteroids.space/data/moons/textures/S9_Phoebe/Phoebe.
↪jpg?size=max")
open("Phoebe_Grayscale.jpg", "wb").write(response.content)
```

[1]:
```
218918
```

First of all, let's define the occulting body, providing the 3D shape model.

[2]:
```
from sora import Body
phoebe = Body(name="Phoebe", shape="Phoebe_Gaskell_50k_poly.obj", ephem="horizons")
# Now, we include the texture in the shape of Phoebe
phoebe.shape.texture = "Phoebe_Grayscale.jpg"
```

```
SORA version: 0.3
```

Now, let's define the occultation parameters using the Gaia-DR3 database

[3]:
```
from sora import Occultation
occ = Occultation(body=phoebe, star="17 31 03.03926 -22 00 58.09752", time='2017-07-06␣
↪16:04:09.020')
```

```
1 GaiaDR3 star found band={'G': 10.334186}
star coordinate at J2016.0: RA=17h31m03.03930s +/- 0.1084 mas, DEC=-22d00m58.0798s +/- 0.
↪0783 mas

Downloading star parameters from I/297/out
```

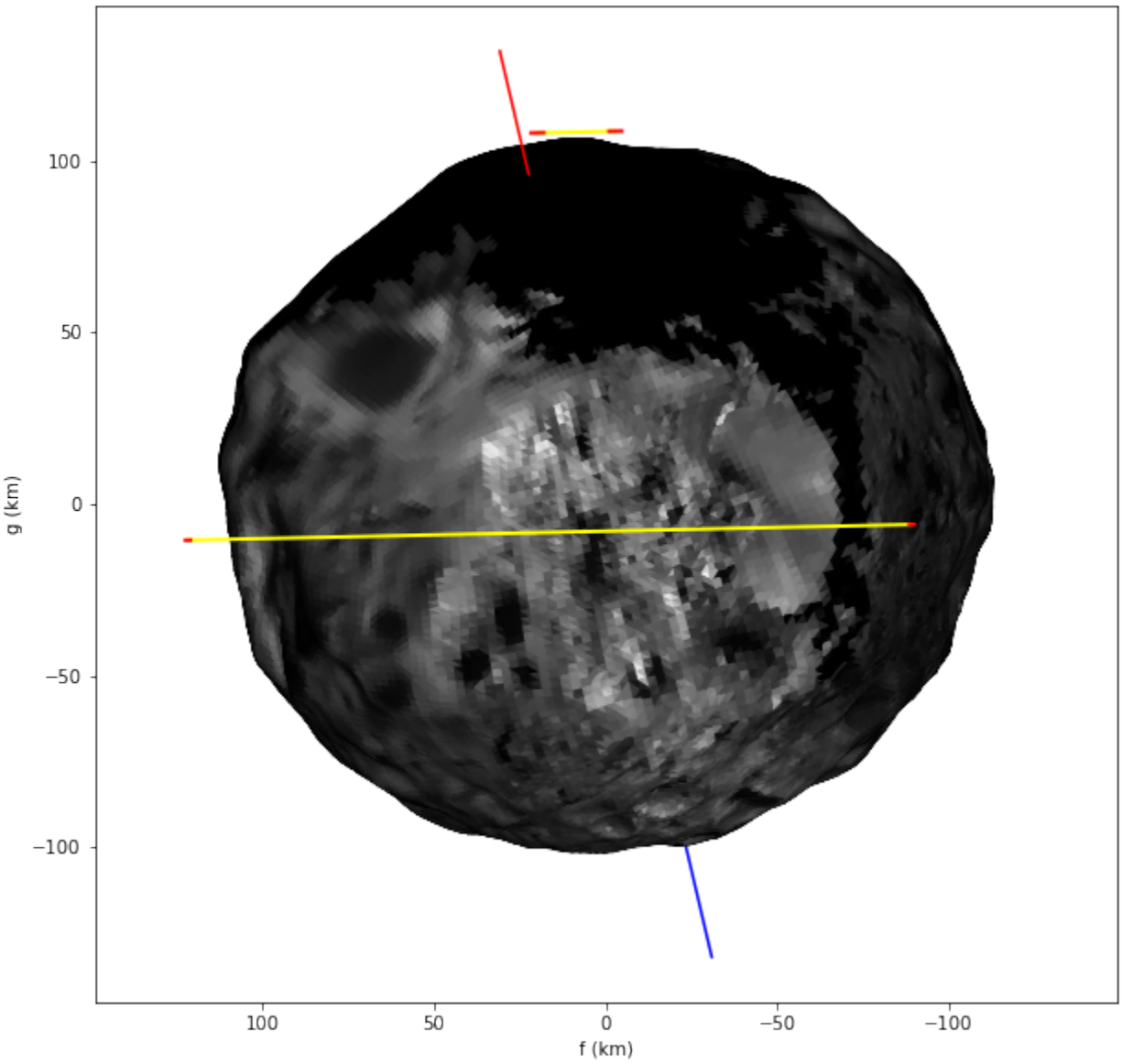
Finally, let's include the observations. These values come directly from Gomes-Júnior et al. (2020).

[4]:
```
from sora import Observer, LightCurve
# The observers
obs1 = Observer(name='Hamamatsu', lon='137 44 23', lat='34 43 07.0', height=17)
obs2 = Observer(name='Miharu', lon='140 26 04.2', lat='37 25 36.7', height=274)
# The "lightcurves"
lc1 = LightCurve(name='Hamamatsu LC', immersion='2017-07-06 16:03:59.55', immersion_
↪err=0.04, emersion='2017-07-06 16:04:11.03', emersion_err=0.04)
lc2 = LightCurve(name='Miharu LC', immersion='2017-07-06 16:04:00.83', immersion_err=0.1,
↪ emersion='2017-07-06 16:04:02.07', emersion_err=0.1)
# Combining them into occultation chords
c1 = occ.chords.add_chord(name='Hamamatsu', observer=obs1, lightcurve=lc1)
c2 = occ.chords.add_chord(name='Miharu', observer=obs2, lightcurve=lc2)
```

Let's look how the occultation chords are projected together with the expected apparent shape of Phoebe at the moment of occultation

[5]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
phoebe.plot(time=occ.tca)
occ.chords.plot_chords(segment='positive', color='yellow')
occ.chords.plot_chords(segment='error', color='red')
plt.xlim(200,-200)
```

[5]:
```
(200.0, -200.0)
```



Important note: When plotting the object apparent shape, we only need to provide the time if the `frame` attribute is defined. In the case of Phoebe, it is automaticaly defined using the Archinal et al. (2018) parameters. Please look at the `frame` attribute in the body guideline.

To see the orientation for a given epoch, use the following method:

[6]:
```
phoebe.get_orientation(time=occ.tca)
```

[6]:
```
{'sub_observer': '332.494 22.3667',
 'sub_solar': '332.493 22.3668',
 'pole_position_angle': <Quantity 13.17471577 deg>,
 'pole_aperture_angle': <Quantity 22.36671143 deg>}
```

The plot shows the occultation chords does not fit well with the projected shape and position, which means an error in ephemeris offset and/or in orientation.

The first approach can be assuming an error in ephemeris offset only. In this case, we do not need to change the orientation. Let's fit the shape providing a range of offsets:

[7]:
```
res1 = occ.fit_shape(center_f=30, dcenter_f=30, center_g=30, dcenter_g=30, loop=50000)
```

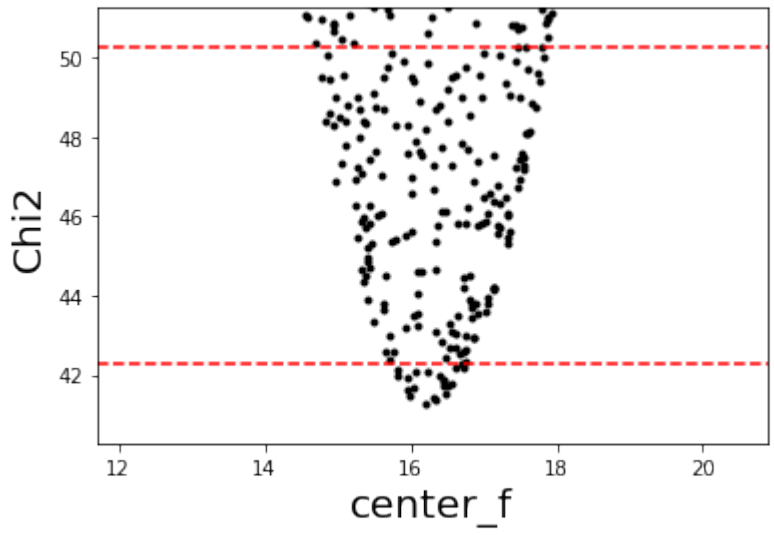
```
100%|| 50000/50000 [01:04<00:00, 775.37it/s]
```

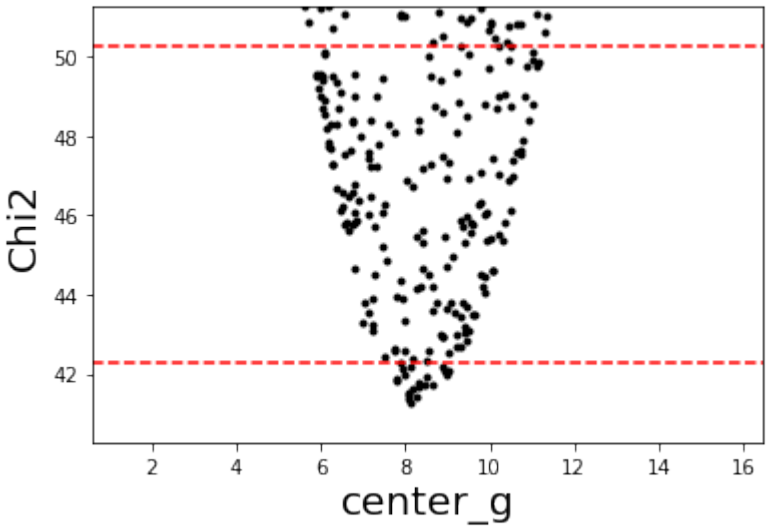
Which results are:

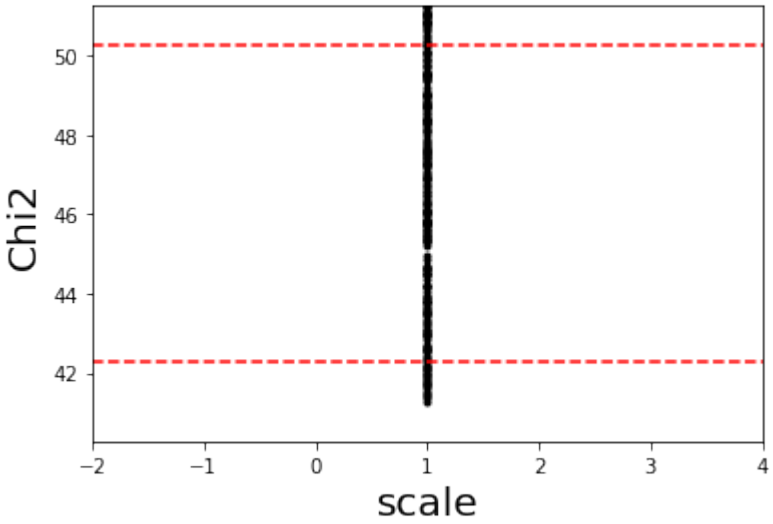
[8]:
```
print(res1)
```

```
Minimum chi-square: 41.268
Number of fitted points: 4
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 20.634

center_f:
    1-sigma: 16.267 +/- 0.461
    3-sigma: 16.295 +/- 1.530

center_g:
    1-sigma: 8.409 +/- 0.623
    3-sigma: 8.522 +/- 2.647

scale:
    1-sigma: 1.000 +/- 0.000
    3-sigma: 1.000 +/- 0.000
```

[9]:
```
res1.plot_chi2()
```



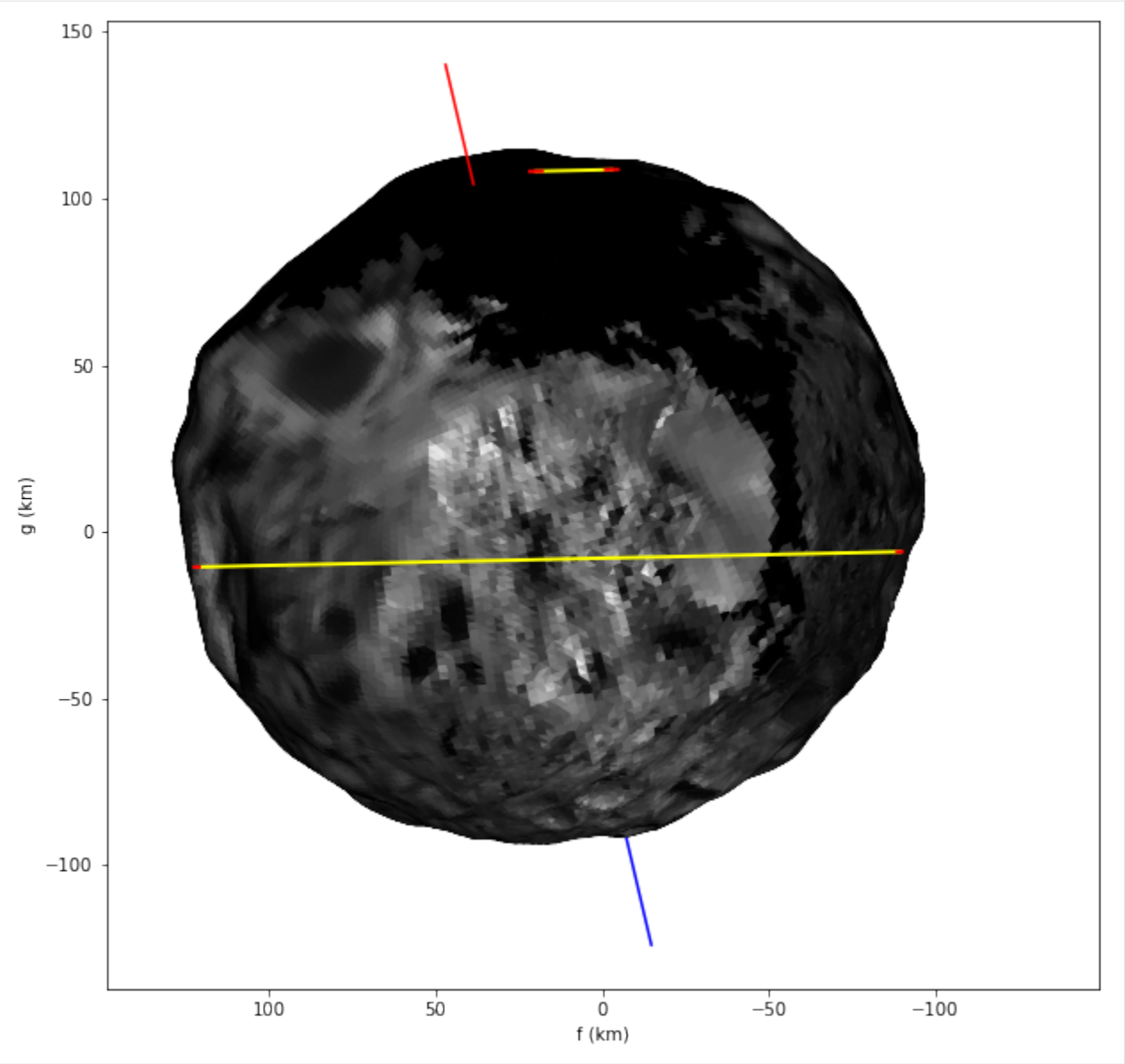




Now, let's redo the plot and see if the occultation can now fit the shape.

[10]:
```
plt.figure(figsize=(10,10))
phoebe.plot(time=occ.tca, **res1.get_values()) # Note the results are unzipped directly␣
↪to the plot function of the body
occ.chords.plot_chords(segment='positive', color='yellow')
occ.chords.plot_chords(segment='error', color='red')
plt.xlim(200,-200)
```

[10]:
```
(200.0, -200.0)
```



We can note the occultation chords still cannot fit well the shape, so probably it may also have an error in orientation.

Important note: SORA can also fit the scale of the 3D shape model. However, Phoebe was observed by Cassini, so its size is very well determined.

Important note: SORA does not have a function to fit the orientation by itself due to the large amount of time needed to characterize with precision the limb of the 3d shape object. So we must do it ourselves, looping throught the orientation. The following cells are an example on how to do this kind of fitting. We must use the function `fit_to_limb` available at `sora.occultation.fitting`.

First, let's look at the expected orientation of Phoebe at the occultation epoch, present in cell 6 above. Then, we'll change only the longitudes, since the direction of the pole is well defined.

[11]:
```
from sora.occultation.fitting import fit_to_limb
import numpy as np
```

(continues on next page)

(continued from previous page)

```
# necessary command to get the contact points of the occultation
name_chord, fg, error = occ.chords.get_limb_points()

def fit_to_lon(lon):
    sub_observer = f"{lon} 22.4"
    limb = phoebe.shape.get_limb(sub_observer=sub_observer, pole_position_angle=13.2)

    # the number of loops is small as we want to first find the global minimum region
    partial_chi2 = fit_to_limb(limb=limb, fg=fg, error=error, center_f=30, dcenter_f=30,␣
↪center_g=30, dcenter_g=30, loop=6000)

    # Includes the longitude value in the chi2 object.
    partial_chi2.data['lon'] = np.repeat(lon, len(partial_chi2))
    partial_chi2._names.append('lon')

    return partial_chi2

# looping through all longitudes with a step of two degrees. The sum function will just␣
↪combine all partial results into one ChiSquare object.
chi2 = np.sum([fit_to_lon(lon) for lon in range(0,360,2)])
```

```
100%|| 6000/6000 [00:10<00:00, 581.33it/s]
100%|| 6000/6000 [00:09<00:00, 649.41it/s]
100%|| 6000/6000 [00:11<00:00, 534.46it/s]
100%|| 6000/6000 [00:09<00:00, 605.92it/s]
100%|| 6000/6000 [00:08<00:00, 696.61it/s]
100%|| 6000/6000 [00:09<00:00, 613.95it/s]
100%|| 6000/6000 [00:09<00:00, 644.41it/s]
100%|| 6000/6000 [00:09<00:00, 617.39it/s]
100%|| 6000/6000 [00:09<00:00, 612.44it/s]
100%|| 6000/6000 [00:08<00:00, 727.27it/s]
100%|| 6000/6000 [00:08<00:00, 704.54it/s]
100%|| 6000/6000 [00:08<00:00, 676.40it/s]
100%|| 6000/6000 [00:08<00:00, 722.92it/s]
100%|| 6000/6000 [00:09<00:00, 636.22it/s]
100%|| 6000/6000 [00:07<00:00, 752.51it/s]
100%|| 6000/6000 [00:09<00:00, 656.77it/s]
100%|| 6000/6000 [00:08<00:00, 716.84it/s]
100%|| 6000/6000 [00:07<00:00, 759.09it/s]
100%|| 6000/6000 [00:09<00:00, 622.67it/s]
100%|| 6000/6000 [00:09<00:00, 629.58it/s]
100%|| 6000/6000 [00:09<00:00, 636.93it/s]
100%|| 6000/6000 [00:07<00:00, 770.88it/s]
100%|| 6000/6000 [00:08<00:00, 674.53it/s]
100%|| 6000/6000 [00:09<00:00, 651.18it/s]
100%|| 6000/6000 [00:08<00:00, 713.16it/s]
100%|| 6000/6000 [00:10<00:00, 576.61it/s]
100%|| 6000/6000 [00:09<00:00, 661.65it/s]
100%|| 6000/6000 [00:09<00:00, 617.73it/s]
100%|| 6000/6000 [00:07<00:00, 782.00it/s]
100%|| 6000/6000 [00:09<00:00, 654.03it/s]
100%|| 6000/6000 [00:07<00:00, 805.79it/s]
```

(continues on next page)

(continued from previous page)

```
100%|| 6000/6000 [00:08<00:00, 680.87it/s]
100%|| 6000/6000 [00:10<00:00, 550.49it/s]
100%|| 6000/6000 [00:10<00:00, 576.17it/s]
100%|| 6000/6000 [00:09<00:00, 655.74it/s]
100%|| 6000/6000 [00:09<00:00, 624.15it/s]
100%|| 6000/6000 [00:08<00:00, 668.96it/s]
100%|| 6000/6000 [00:10<00:00, 592.09it/s]
100%|| 6000/6000 [00:08<00:00, 678.54it/s]
100%|| 6000/6000 [00:07<00:00, 753.47it/s]
100%|| 6000/6000 [00:09<00:00, 663.00it/s]
100%|| 6000/6000 [00:10<00:00, 590.75it/s]
100%|| 6000/6000 [00:09<00:00, 645.11it/s]
100%|| 6000/6000 [00:10<00:00, 586.89it/s]
100%|| 6000/6000 [00:09<00:00, 657.28it/s]
100%|| 6000/6000 [00:08<00:00, 680.53it/s]
100%|| 6000/6000 [00:09<00:00, 656.93it/s]
100%|| 6000/6000 [00:09<00:00, 649.28it/s]
100%|| 6000/6000 [00:09<00:00, 602.70it/s]
100%|| 6000/6000 [00:09<00:00, 637.84it/s]
100%|| 6000/6000 [00:09<00:00, 638.21it/s]
100%|| 6000/6000 [00:08<00:00, 688.26it/s]
100%|| 6000/6000 [00:08<00:00, 684.12it/s]
100%|| 6000/6000 [00:09<00:00, 657.50it/s]
100%|| 6000/6000 [00:10<00:00, 555.74it/s]
100%|| 6000/6000 [00:09<00:00, 615.50it/s]
100%|| 6000/6000 [00:09<00:00, 640.73it/s]
100%|| 6000/6000 [00:09<00:00, 627.79it/s]
100%|| 6000/6000 [00:09<00:00, 641.70it/s]
100%|| 6000/6000 [00:09<00:00, 633.32it/s]
100%|| 6000/6000 [00:08<00:00, 689.94it/s]
100%|| 6000/6000 [00:08<00:00, 698.51it/s]
100%|| 6000/6000 [00:08<00:00, 689.16it/s]
100%|| 6000/6000 [00:09<00:00, 603.39it/s]
100%|| 6000/6000 [00:09<00:00, 661.19it/s]
100%|| 6000/6000 [00:09<00:00, 635.45it/s]
100%|| 6000/6000 [00:08<00:00, 692.84it/s]
100%|| 6000/6000 [00:10<00:00, 585.37it/s]
100%|| 6000/6000 [00:09<00:00, 632.99it/s]
100%|| 6000/6000 [00:08<00:00, 745.53it/s]
100%|| 6000/6000 [00:07<00:00, 786.97it/s]
100%|| 6000/6000 [00:11<00:00, 540.80it/s]
100%|| 6000/6000 [00:08<00:00, 689.39it/s]
100%|| 6000/6000 [00:10<00:00, 553.06it/s]
100%|| 6000/6000 [00:09<00:00, 623.70it/s]
100%|| 6000/6000 [00:09<00:00, 632.45it/s]
100%|| 6000/6000 [00:08<00:00, 702.03it/s]
100%|| 6000/6000 [00:09<00:00, 615.36it/s]
100%|| 6000/6000 [00:08<00:00, 698.08it/s]
100%|| 6000/6000 [00:08<00:00, 671.20it/s]
100%|| 6000/6000 [00:07<00:00, 761.48it/s]
100%|| 6000/6000 [00:08<00:00, 673.90it/s]
100%|| 6000/6000 [00:09<00:00, 641.89it/s]
```

(continues on next page)

(continued from previous page)

```
100%|| 6000/6000 [00:08<00:00, 685.80it/s]
100%|| 6000/6000 [00:07<00:00, 759.82it/s]
100%|| 6000/6000 [00:08<00:00, 672.61it/s]
100%|| 6000/6000 [00:08<00:00, 725.74it/s]
100%|| 6000/6000 [00:08<00:00, 676.80it/s]
100%|| 6000/6000 [00:10<00:00, 582.13it/s]
100%|| 6000/6000 [00:09<00:00, 652.15it/s]
100%|| 6000/6000 [00:08<00:00, 719.87it/s]
100%|| 6000/6000 [00:08<00:00, 724.76it/s]
100%|| 6000/6000 [00:08<00:00, 675.08it/s]
100%|| 6000/6000 [00:09<00:00, 657.33it/s]
100%|| 6000/6000 [00:08<00:00, 679.02it/s]
100%|| 6000/6000 [00:08<00:00, 734.30it/s]
100%|| 6000/6000 [00:08<00:00, 678.17it/s]
100%|| 6000/6000 [00:08<00:00, 745.24it/s]
100%|| 6000/6000 [00:08<00:00, 699.26it/s]
100%|| 6000/6000 [00:08<00:00, 749.33it/s]
100%|| 6000/6000 [00:08<00:00, 704.84it/s]
100%|| 6000/6000 [00:08<00:00, 701.54it/s]
100%|| 6000/6000 [00:09<00:00, 641.50it/s]
100%|| 6000/6000 [00:09<00:00, 652.40it/s]
100%|| 6000/6000 [00:07<00:00, 777.66it/s]
100%|| 6000/6000 [00:08<00:00, 718.92it/s]
100%|| 6000/6000 [00:08<00:00, 712.02it/s]
100%|| 6000/6000 [00:07<00:00, 750.29it/s]
100%|| 6000/6000 [00:08<00:00, 698.21it/s]
100%|| 6000/6000 [00:08<00:00, 686.22it/s]
100%|| 6000/6000 [00:08<00:00, 738.56it/s]
100%|| 6000/6000 [00:09<00:00, 657.67it/s]
100%|| 6000/6000 [00:08<00:00, 716.25it/s]
100%|| 6000/6000 [00:08<00:00, 695.68it/s]
100%|| 6000/6000 [00:09<00:00, 614.93it/s]
100%|| 6000/6000 [00:09<00:00, 609.50it/s]
100%|| 6000/6000 [00:09<00:00, 665.67it/s]
100%|| 6000/6000 [00:09<00:00, 659.98it/s]
100%|| 6000/6000 [00:07<00:00, 802.97it/s]
100%|| 6000/6000 [00:08<00:00, 698.85it/s]
100%|| 6000/6000 [00:09<00:00, 614.89it/s]
100%|| 6000/6000 [00:09<00:00, 633.67it/s]
100%|| 6000/6000 [00:07<00:00, 803.76it/s]
100%|| 6000/6000 [00:08<00:00, 736.84it/s]
100%|| 6000/6000 [00:09<00:00, 640.88it/s]
100%|| 6000/6000 [00:09<00:00, 648.59it/s]
100%|| 6000/6000 [00:08<00:00, 713.97it/s]
100%|| 6000/6000 [00:08<00:00, 734.06it/s]
100%|| 6000/6000 [00:08<00:00, 695.05it/s]
100%|| 6000/6000 [00:08<00:00, 726.26it/s]
100%|| 6000/6000 [00:09<00:00, 650.43it/s]
100%|| 6000/6000 [00:09<00:00, 665.66it/s]
100%|| 6000/6000 [00:09<00:00, 639.03it/s]
100%|| 6000/6000 [00:09<00:00, 613.13it/s]
100%|| 6000/6000 [00:08<00:00, 703.28it/s]
```

(continues on next page)

(continued from previous page)

```
100%|| 6000/6000 [00:10<00:00, 566.46it/s]
100%|| 6000/6000 [00:08<00:00, 672.13it/s]
100%|| 6000/6000 [00:09<00:00, 638.73it/s]
100%|| 6000/6000 [00:08<00:00, 712.23it/s]
100%|| 6000/6000 [00:09<00:00, 665.15it/s]
100%|| 6000/6000 [00:07<00:00, 753.53it/s]
100%|| 6000/6000 [00:08<00:00, 746.39it/s]
100%|| 6000/6000 [00:07<00:00, 782.83it/s]
100%|| 6000/6000 [00:08<00:00, 741.09it/s]
100%|| 6000/6000 [00:08<00:00, 710.01it/s]
100%|| 6000/6000 [00:08<00:00, 706.13it/s]
100%|| 6000/6000 [00:08<00:00, 719.60it/s]
100%|| 6000/6000 [00:08<00:00, 717.94it/s]
100%|| 6000/6000 [00:08<00:00, 723.18it/s]
100%|| 6000/6000 [00:07<00:00, 772.13it/s]
100%|| 6000/6000 [00:08<00:00, 682.55it/s]
100%|| 6000/6000 [00:08<00:00, 687.36it/s]
100%|| 6000/6000 [00:10<00:00, 598.40it/s]
100%|| 6000/6000 [00:10<00:00, 595.49it/s]
100%|| 6000/6000 [00:08<00:00, 708.57it/s]
100%|| 6000/6000 [00:09<00:00, 618.93it/s]
100%|| 6000/6000 [00:08<00:00, 690.32it/s]
100%|| 6000/6000 [00:08<00:00, 668.71it/s]
100%|| 6000/6000 [00:09<00:00, 658.81it/s]
100%|| 6000/6000 [00:08<00:00, 674.74it/s]
100%|| 6000/6000 [00:09<00:00, 648.03it/s]
100%|| 6000/6000 [00:10<00:00, 564.91it/s]
100%|| 6000/6000 [00:09<00:00, 628.87it/s]
100%|| 6000/6000 [00:08<00:00, 720.31it/s]
100%|| 6000/6000 [00:08<00:00, 735.56it/s]
100%|| 6000/6000 [00:09<00:00, 641.69it/s]
100%|| 6000/6000 [00:08<00:00, 676.46it/s]
100%|| 6000/6000 [00:07<00:00, 751.05it/s]
100%|| 6000/6000 [00:09<00:00, 660.96it/s]
100%|| 6000/6000 [00:08<00:00, 719.34it/s]
100%|| 6000/6000 [00:09<00:00, 664.53it/s]
100%|| 6000/6000 [00:07<00:00, 827.19it/s]
100%|| 6000/6000 [00:08<00:00, 723.41it/s]
100%|| 6000/6000 [00:08<00:00, 713.85it/s]
100%|| 6000/6000 [00:08<00:00, 713.96it/s]
100%|| 6000/6000 [00:08<00:00, 694.20it/s]
100%|| 6000/6000 [00:08<00:00, 676.51it/s]
100%|| 6000/6000 [00:09<00:00, 612.55it/s]
100%|| 6000/6000 [00:08<00:00, 711.44it/s]
100%|| 6000/6000 [00:08<00:00, 669.35it/s]
```

[12]:
```
print(chi2)
```

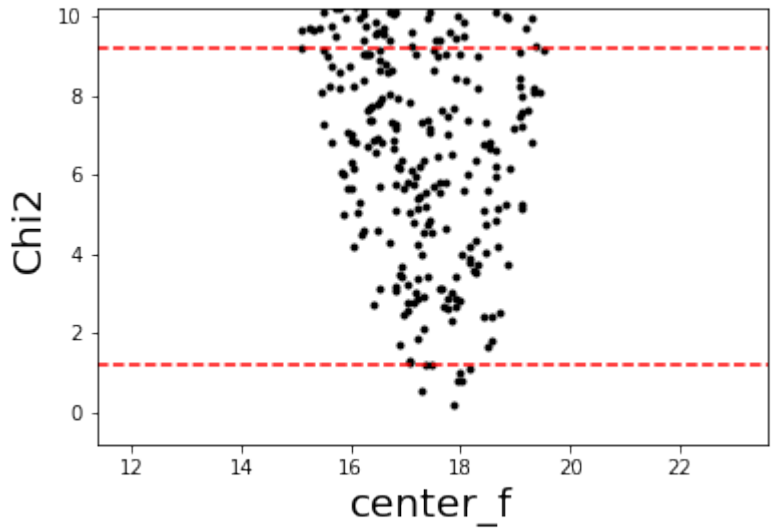
```
Minimum chi-square: 0.184
Number of fitted points: 4
Number of fitted parameters: 3
Minimum chi-square per degree of freedom: 0.184
```

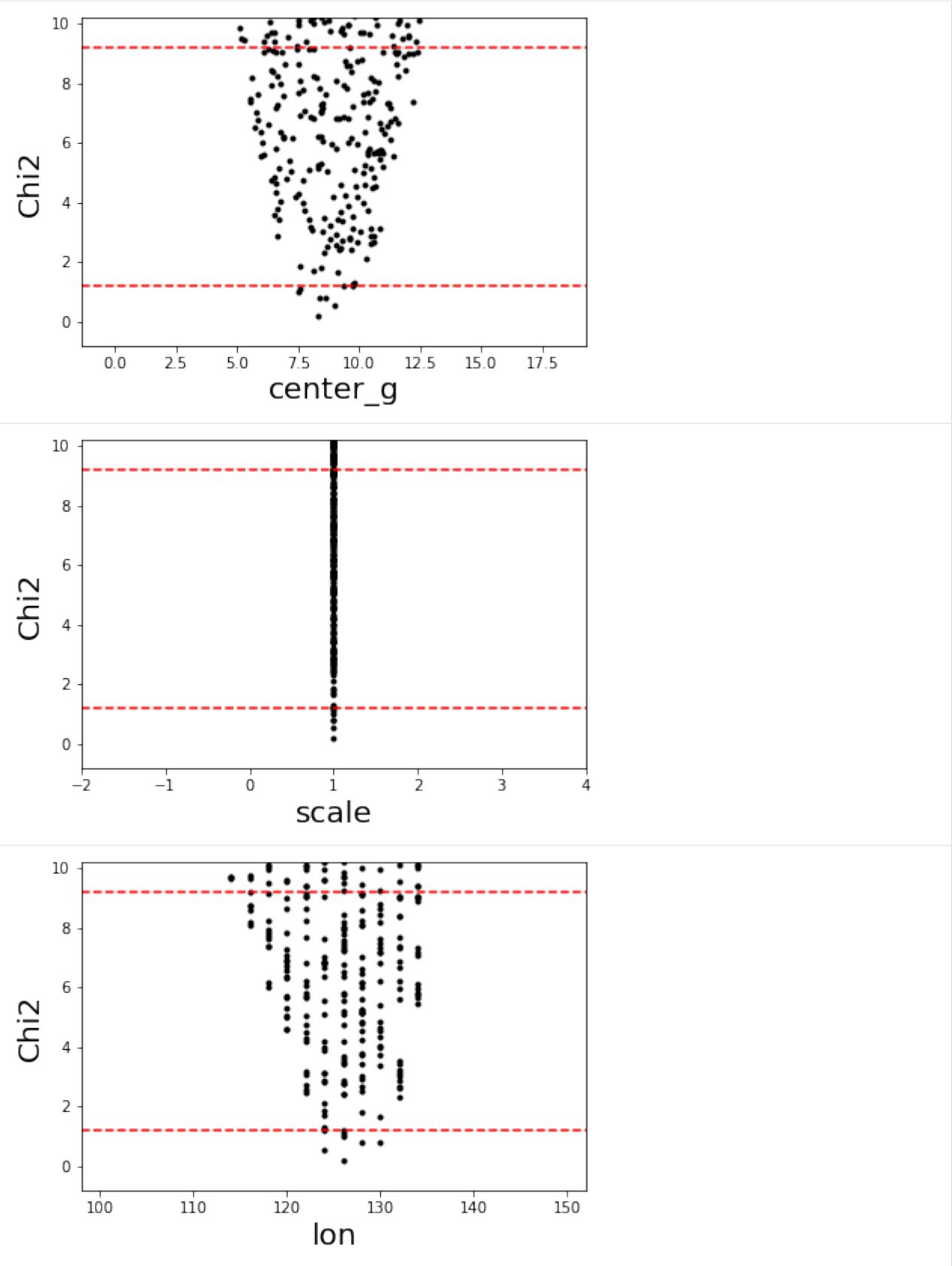
(continues on next page)

(continued from previous page)

```
center_f:
    1-sigma: 17.716 +/- 0.436
    3-sigma: 17.483 +/- 2.038

center_g:
    1-sigma: 8.265 +/- 0.753
    3-sigma: 8.959 +/- 3.443

scale:
    1-sigma: 1.000 +/- 0.000
    3-sigma: 1.000 +/- 0.000

lon:
    1-sigma: 127.000 +/- 3.000
    3-sigma: 125.000 +/- 9.000
```

[13]:
```
chi2.plot_chi2()
```



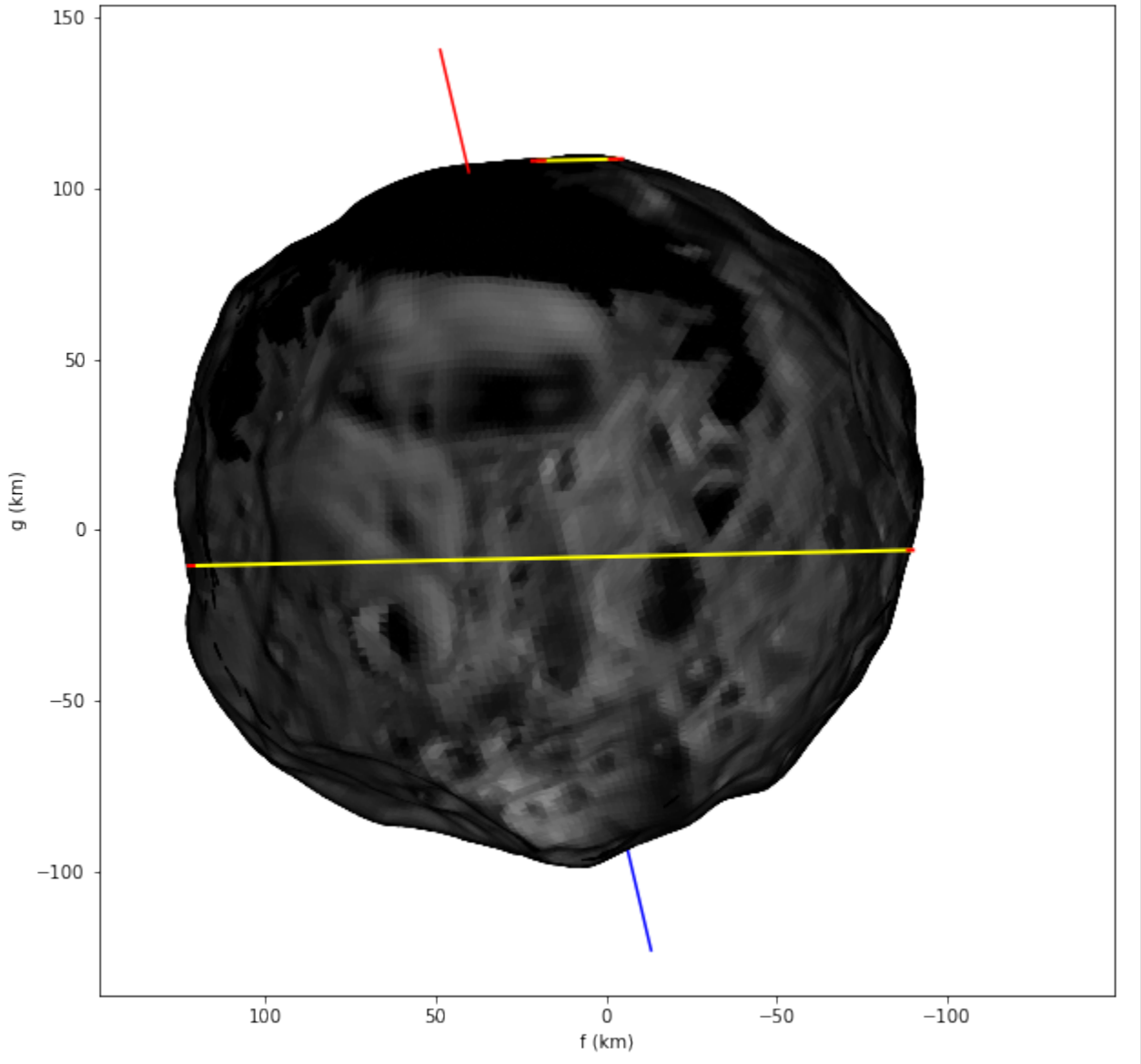

Chi2
center_g
Chi2
scale
Chi2
lon

Now we can plot the occultation again with the new orientation

[14]:
```
plt.figure(figsize=(10,10))
phoebe.plot(sub_observer="123.0 22.4", pole_position_angle=13.2, center_f=17.8, center_
↪g=8.7)
occ.chords.plot_chords(segment='positive', color='yellow')
occ.chords.plot_chords(segment='error', color='red')
plt.xlim(200,-200)
```

```
/home/altair/Documentos/códigos/SORA/sora/body/core.py:444: UserWarning: No time is␣
↪giving or frame is not defined. Plotting without computing orientation. To provide␣
↪orientation, please plot from shape directly.
  warnings.warn('No time is giving or frame is not defined. Plotting without computing␣
↪orientation. '
```

[14]:
```
(200.0, -200.0)
```



The occultation chords fit much better the 3D shape now. To have better values, we should redo the fitting process

using smaller steps in longitude.

And the final astrometric position is:

[15]:
```
occ.new_astrometric_position(time=occ.tca, offset=[17.8, 8.7, 'km'], error=[0.5, 1.0, 'km
↪'])
```

```
Ephemeris offset (km): X = 17.8 km +/- 0.5 km; Y = 8.7 km +/- 1.0 km
Ephemeris offset (mas): da_cos_dec = 2.700 +/- 0.076; d_dec = 1.319 +/- 0.152

Astrometric object position at time 2017-07-06 16:07:09.000 for reference 'geocenter'
RA = 17 31 03.0403552 +/- 0.203 mas; DEC = -22 00 57.352765 +/- 0.238 mas
```

We should notice in Gomes-Júnior et al. (2020) the analysis was pre-SORA, and the catalogue used was Gaia-DR2, so there may be differences in the result found in this example and the published one.

8.5 Using optimization fitting routines (since v0.3)

New optimization algorithms are introduced in SORA v0.3 to allow faster convergence to the solution when modeling a light curve or fitting an ellipse to the positive chords.

Besides the Monte Carlo technique to find the best solution, there are now two optimization methods available: one, the default option, based on dumped least-squares (Levenberg-Marquardt), and the other based on genetic algorithms (differential evolution).

To maintain backward compatibility with the chi-square result object the estimate of uncertainties still rely on Monte Carlo simulations, but now with parallel processing available.

Two new keywords were added to `sora.LightCurve.occ_lcfit` and to `sora.Occultation.fit_ellipse` functions: `"method"` and `"threads"`. The first allows the optimization method selection, while the latter sets the number of parallel processes when using multithreading. The parallel processing relies on the multiprocessing python module, which is built on shared memory parallel processing.

Methods available:

- *chisqr*: Monte Carlo computation method used in versions of SORA <= 0.2.1.
- *fastchi*: Monte Carlo computation method, allows multithreading. Used with keyword “threads”.
- *least_squares* or *ls*: uses Levenberg-Marquardt as optimization method.
- *differential_evolution* or *de*: uses genetic algorithms as optimization method.

8.5.1 Determining the instants of a light curve using optimization

Let’s use the examples available in the Getting Started guide. In the following examples, we will always set the parameter `"threads = 5"`, meaning the number of parallel processes generated during the computations. **It is wise to set the number of threads to a number below the total number of cores available in your CPU (usually n_cores-1) to avoid freezing the computer/interaction during the computations.**

[1]:
```
## SORA package
from sora import Occultation, Body, Star, LightCurve, Observer
from sora.prediction import prediction
from sora.extra import draw_ellipse

## Other main packages
```

(continues on next page)

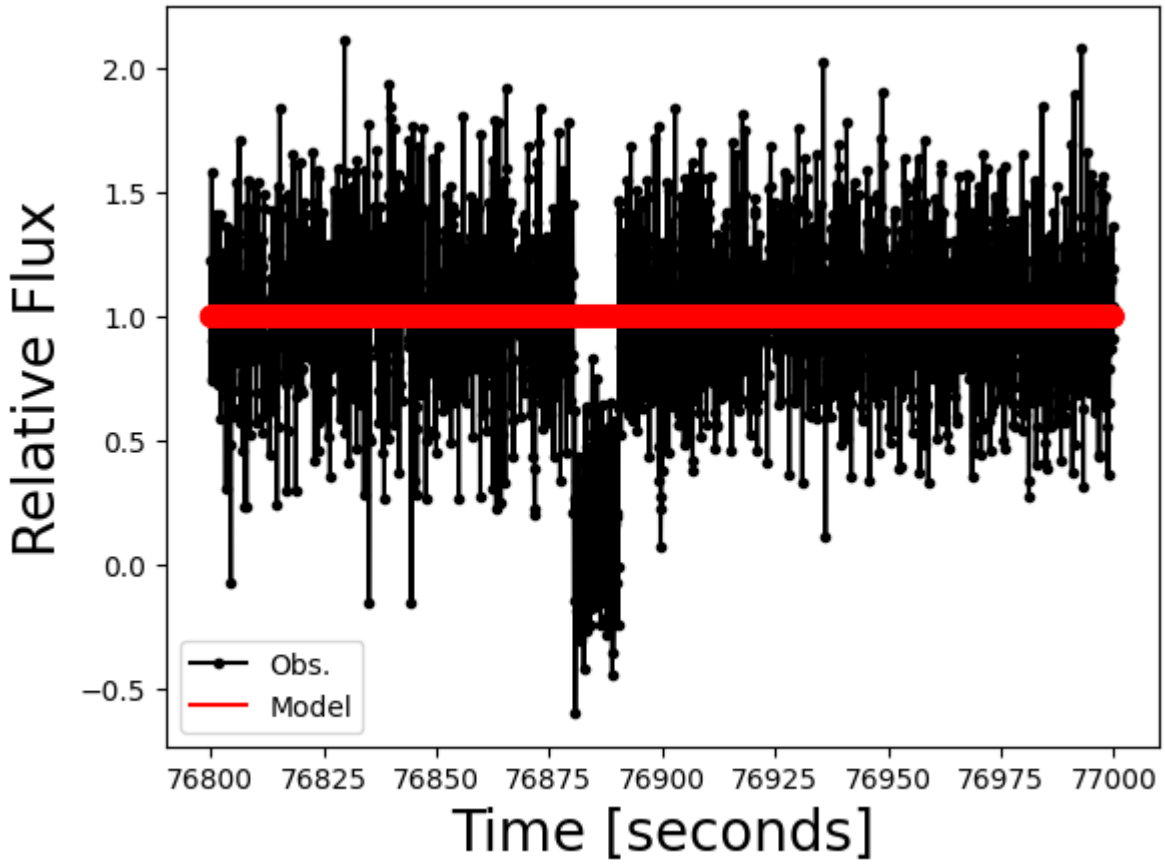
(continued from previous page)

```
from astropy.time import Time
import astropy.units as u

## Usual packages
import numpy as np
import matplotlib.pylab as pl
import os
```

```
SORA version: 0.3
```

[2]:
```
out_lc = LightCurve(name='Outeniqua lc', file='../guidelines/input/lightcurves/lc_
↪example.dat',
                    exptime=0.100, usecols=[0,1])

print('Name: {}'.format(out_lc.name))
print('Initial time: {} UTC'.format(out_lc.initial_time.iso))
print('Mean time:    {} UTC'.format(out_lc.time_mean.iso))
print('End time:     {} UTC'.format(out_lc.end_time.iso))
print('Exposure time: {:.4f} s'.format(out_lc.exptime))
print('Cycle time:    {:.4f} s'.format(out_lc.cycle))

# However, to model a light curve we need to set the shadow/object velocity, object's␣
↪distance,
# and the star apparend diameter. Let's set some arbitrary values for the sake of example.

out_lc.set_vel(vel=22.0)
print('Vel: {:3.1f} km/s'.format(out_lc.vel))

out_lc.set_dist(dist=15)
print('Distance: {:3.1f} AU'.format(out_lc.dist))

print('Observational wavelength centred at {:1.3f} with a bandwidth of {:1.3f} microns'
      .format(out_lc.central_bandpass,out_lc.delta_bandpass))

print('Fresnel Scale: {:1.3f} km'.format(out_lc.fresnel_scale))

out_lc.set_star_diam(d_star=0.2)
print('Stellar diameter: {:1.2f} km'.format(out_lc.d_star))

out_lc.plot_lc()
```

```
Name: Outeniqua lc
Initial time: 2017-06-22 21:20:00.056 UTC
Mean time:    2017-06-22 21:21:40.007 UTC
End time:     2017-06-22 21:23:19.958 UTC
Exposure time: 0.1000 s
Cycle time:    0.1002 s
Vel: 22.0 km/s
Distance: 15.0 AU
Observational wavelength centred at 0.700 with a bandwidth of 0.300 microns
Fresnel Scale: 0.881 km
Stellar diameter: 0.20 km
```



[3]:
```
# lets autodect the main parameters of the occultation
autodetect = out_lc.occ_detect()
[print('{}: {}'.format(key, value)) for key, value in autodetect.items()]

# and from these results set some initial parameters for our fit
immersion_time = autodetect['immersion_time']
emersion_time = autodetect['emersion_time']
delta_t = autodetect['time_err']*5
```

```
rank: 1
occultation_duration: 9.921853244304657
central_time: 76885.2990051359
immersion_time: 76880.33807851374
emersion_time: 76890.25993175805
time_err: 0.05011036992073059
depth: 0.8912593186105263
depth_err: 0.30084243363867164
baseline: 1.0292228832105264
baseline_err: 0.31961412122793376
snr: 2.9625452361583338
occ_mask: [False False False ... False False False]
```

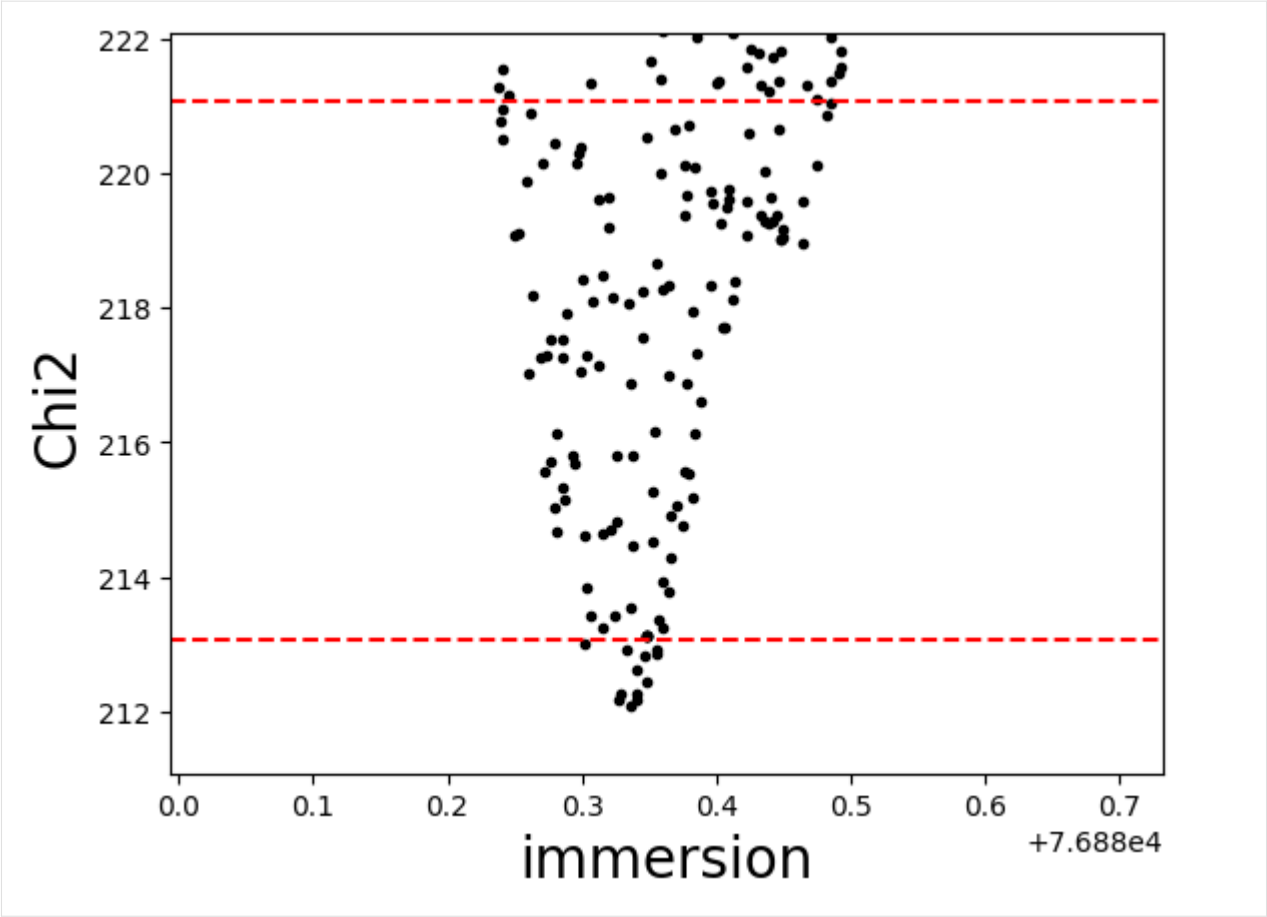
When using optimization methods ("least_squares" or "differential_evolution") a first run can be done with a smaller value for "loop", since we are interested in the best-fit parameters. After a first run, one can better establish the region around where the Monte Carlos simulations will be sampled to produce much better populated χ^2 maps and therefore

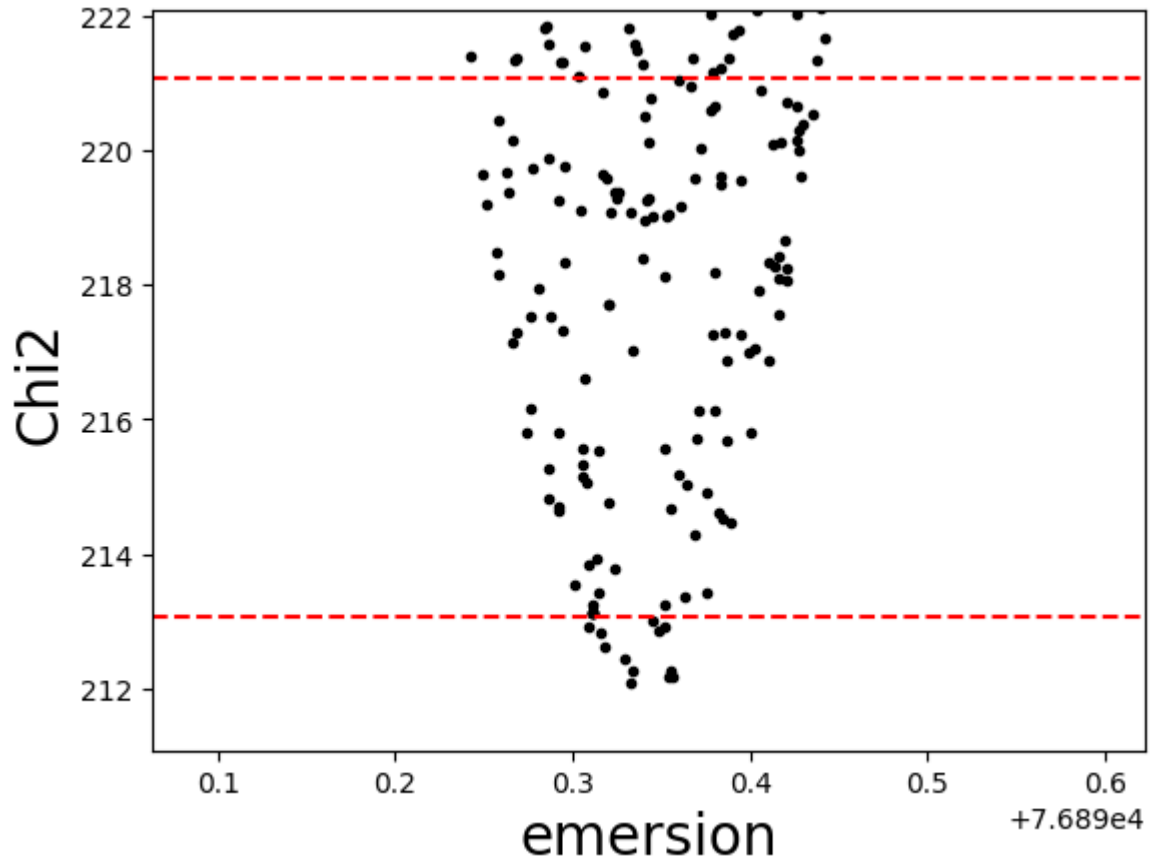
more robust uncertainty estimates.

In this first run, we will focus on finding the best fit. Then, we will increase the loop to explore the parameter space around this best fit first found.

[4]:
```
# Here we will set tmax and tmin to use a subset of the data

tmin = immersion_time  - 5 # seconds relative to tref
tmax = emersion_time   + 5 # seconds relative to tref

out_chi2 = out_lc.occ_lcfit(immersion_time=immersion_time, emersion_time=emersion_time,␣
↪delta_t=delta_t,
                            flux_min=0, flux_max=1, tmax=tmax, tmin=tmin,
                            sigma='auto', sigma_result=1, loop=1000,
                            method='ls', threads=5)
print('\n')
print(out_chi2)
out_chi2.plot_chi2()
```

```
Lightcurve fit: ||  - 100%


Minimum chi-square: 212.079
Number of fitted points: 200
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 1.071

immersion:
    1-sigma: 76880.329 +/- 0.027
    3-sigma: 76880.363 +/- 0.123

emersion:
    1-sigma: 76890.332 +/- 0.024
    3-sigma: 76890.343 +/- 0.093
```


222
220
218
216
214
212
Chi2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
+7.688e4
immersion



Since the χ^2 maps show we have a good coverage around the minimum, we only need to set the parameter "loop" to a higher value in order do produce more confident error bars.

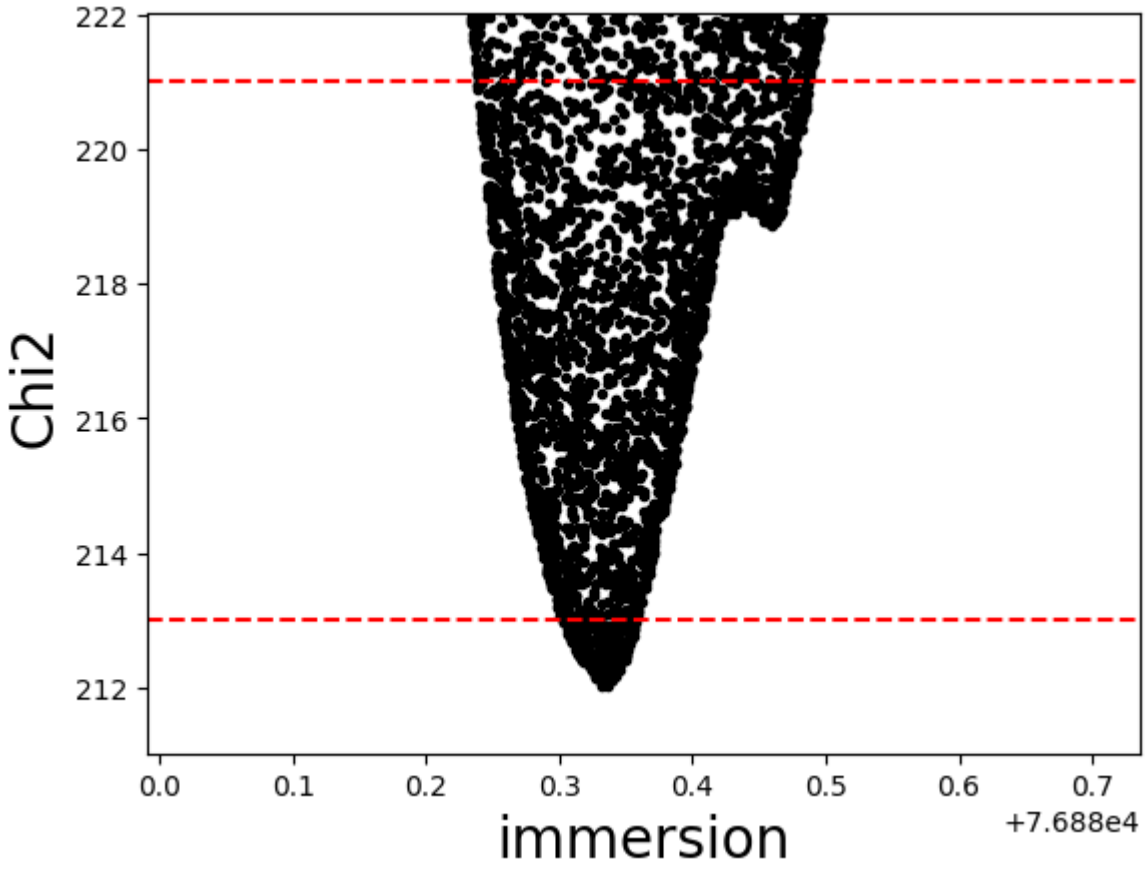
[5]:
```
out_chi2 = out_lc.occ_lcfit(immersion_time=immersion_time, emersion_time=emersion_time,␣
↪delta_t=delta_t,
                            flux_min=0, flux_max=1, tmax=tmax, tmin=tmin,
                            sigma='auto', sigma_result=1, loop=20000,
                            method='ls', threads=5)
print('\n')
print(out_chi2)
out_chi2.plot_chi2()
```

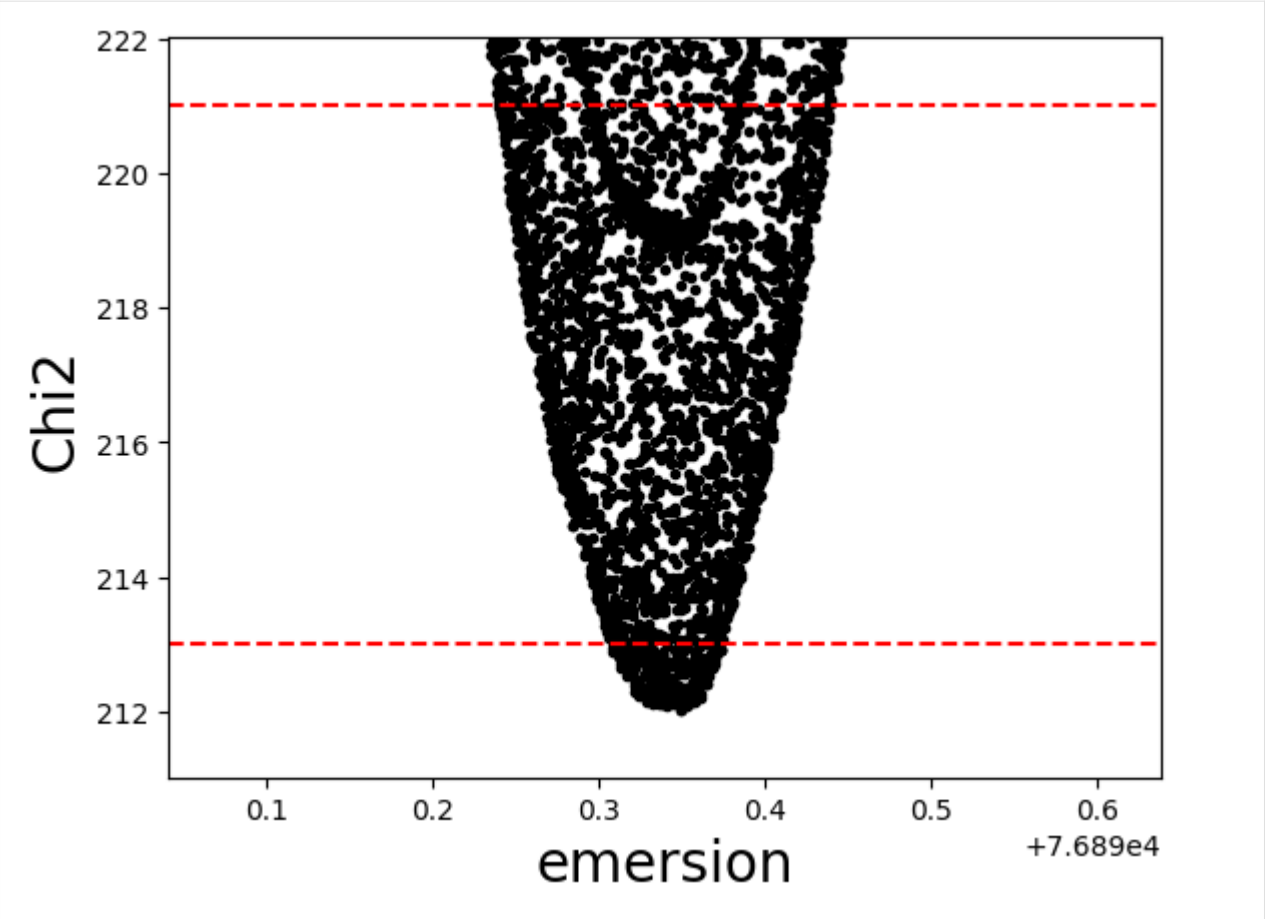
```
Lightcurve fit: ||  - 100%


Minimum chi-square: 212.021
Number of fitted points: 200
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 1.071

immersion:
    1-sigma: 76880.330 +/- 0.027
    3-sigma: 76880.363 +/- 0.124
```

(continues on next page)

(continued from previous page)

```
emersion:
    1-sigma: 76890.341 +/- 0.033
    3-sigma: 76890.339 +/- 0.100
```





And we can produce similar results using the "differential_evolution" or "de" method.

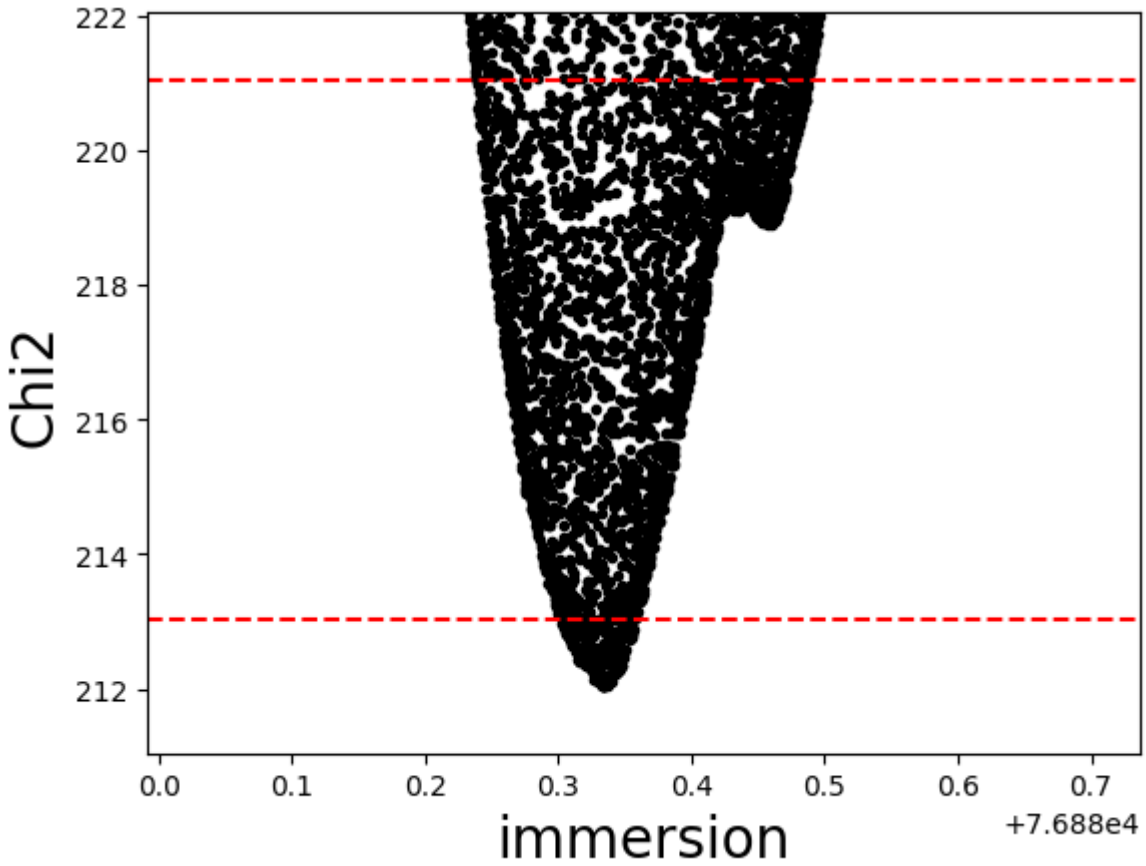
[6]:
```
out_chi2 = out_lc.occ_lcfit(immersion_time=immersion_time, emersion_time=emersion_time,␣
↪delta_t=delta_t,
                            flux_min=0, flux_max=1, tmax=tmax, tmin=tmin,
                            sigma='auto', sigma_result=1, loop=20000,
                            method='de', threads=5)
print('\n')
print(out_chi2)
out_chi2.plot_chi2()
```

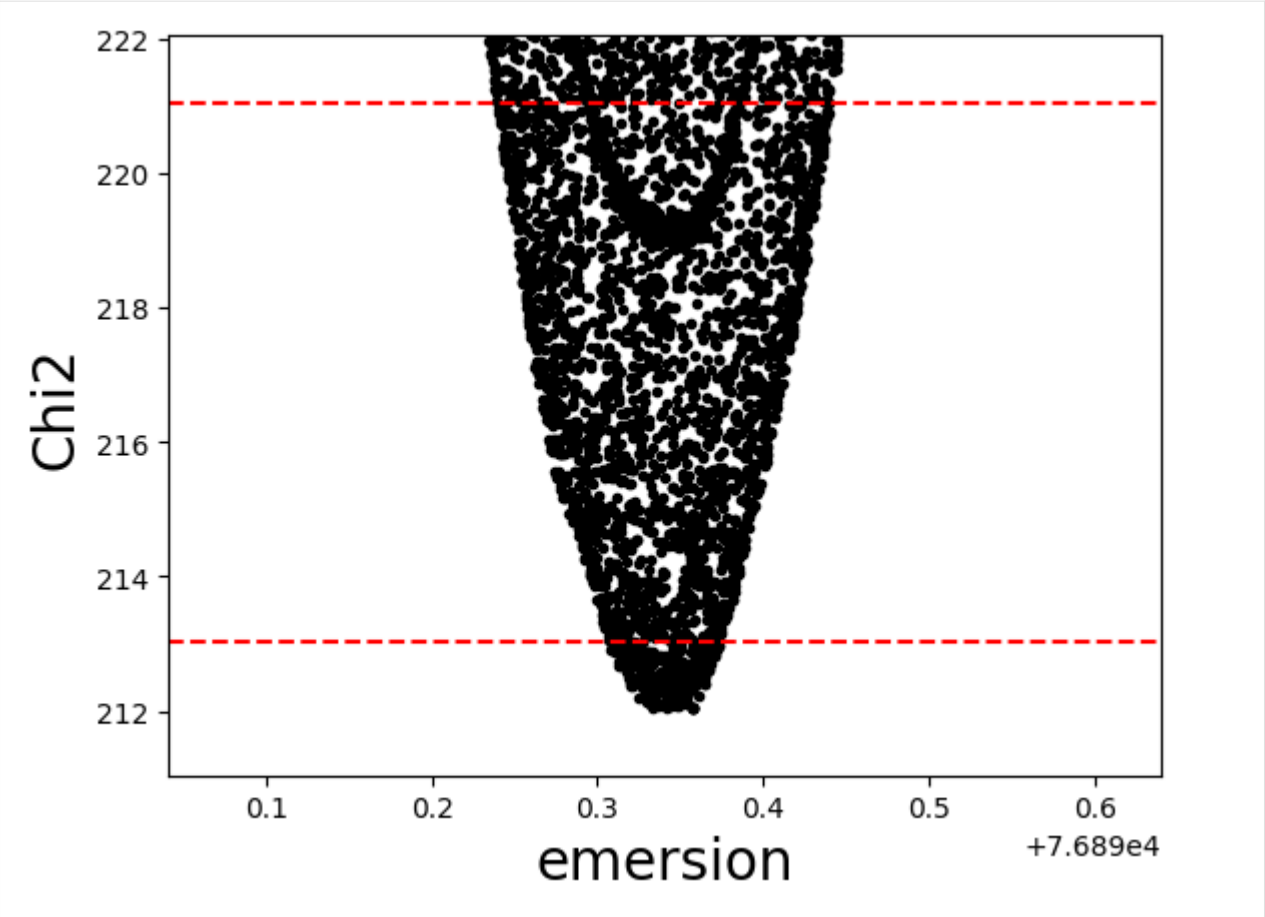
```
Lightcurve fit: ||  - 100%


Minimum chi-square: 212.042
Number of fitted points: 200
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 1.071

immersion:
    1-sigma: 76880.330 +/- 0.027
    3-sigma: 76880.364 +/- 0.124

emersion:
```

(continues on next page)

(continued from previous page)

```
1-sigma: 76890.341 +/- 0.033
3-sigma: 76890.340 +/- 0.100
```





You can always use the "chiqsr" or "fastchi" if one of the above methods fail. Although more time consuming they will ensure to reach the minimum if there is some failure or trapping during convergence of the previous methods.

Let's check a summary of the results obtained with the last fitting procedure.

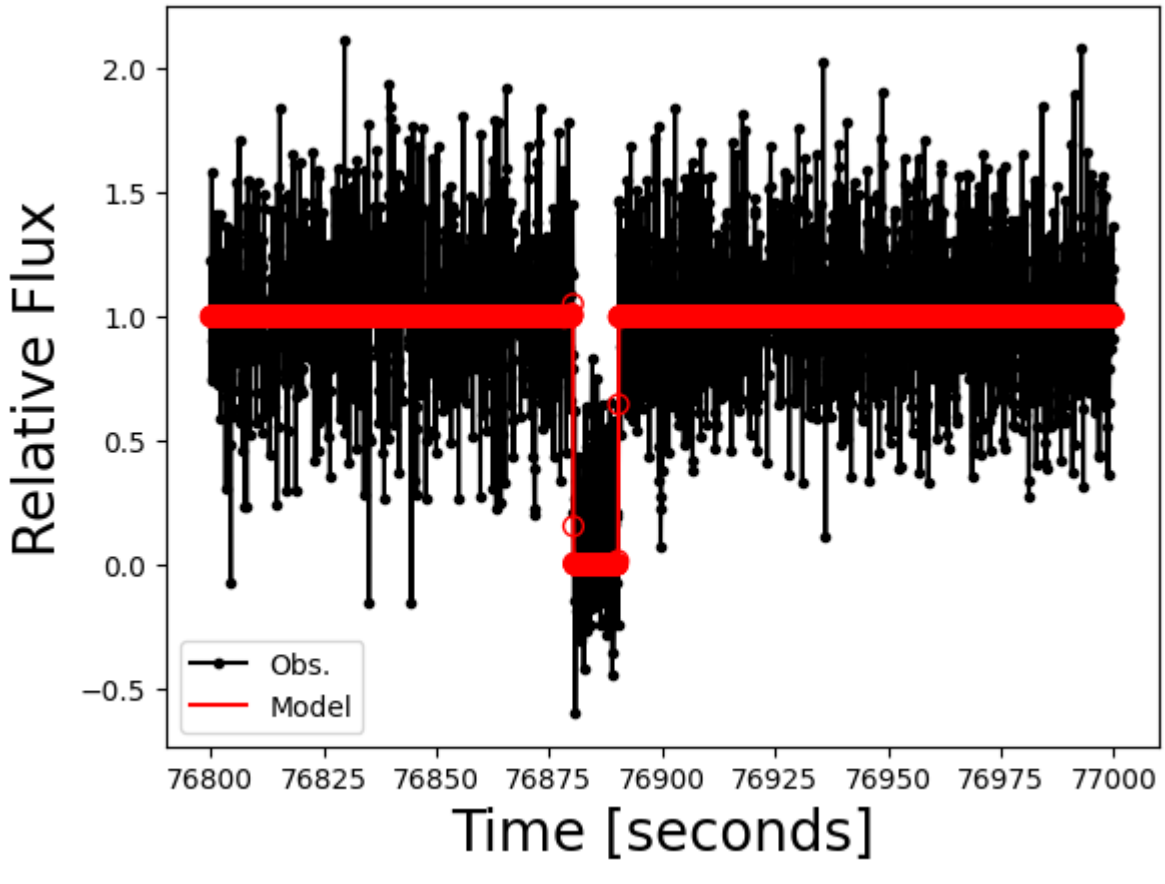
[7]:
```
print(out_lc)
out_lc.plot_lc()
```

```
Light curve name: Outeniqua lc
Initial time: 2017-06-22 21:20:00.056 UTC
End time:     2017-06-22 21:23:19.958 UTC
Duration:     3.332 minutes
Time offset:  0.000 seconds

Exposure time:    0.1000 seconds
Cycle time:       0.1002 seconds
Num. data points: 2000

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      15.00 AU
Used shadow velocity: 22.000 km/s
Fresnel scale:        0.040 seconds or 0.88 km
Stellar size effect:  0.009 seconds or 0.20 km

Object LightCurve model was not fitted.
```

(continues on next page)

(continued from previous page)

```
Immersion time: 2017-06-22 21:21:20.330 UTC +/- 0.027 seconds
Emersion time:  2017-06-22 21:21:30.341 UTC +/- 0.033 seconds

Monte Carlo chi square fit.

Minimum chi-square: 212.042
Number of fitted points: 200
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 1.071

immersion:
    1-sigma: 76880.330 +/- 0.027
    3-sigma: 76880.364 +/- 0.124

emersion:
    1-sigma: 76890.341 +/- 0.033
    3-sigma: 76890.340 +/- 0.100
```



8.5.2 Fitting an ellipse to positive chords using optimization

Again, let's use an examples of the Getting Started guide. In the following examples we will always set the parameter "threads = 5", meaning the number of parallel processes gerenated during the computations. **It is wise to set the number of threads to a number below the total number of cores available in your cpu (usually n_cores-1) to avoid freezing the computer/interaction during the computations.**

[8]:
```
# Let's first set up the occultation event
# and consider an Solar System Body

chariklo = Body(name='Chariklo',
                ephem=['../guidelines/input/bsp/Chariklo.bsp', '../guidelines/input/bsp/
↪de438_small.bsp'])

# draw the predictions for the epoch of interest
pred = prediction(body=chariklo, time_beg='2017-06-20',time_end='2017-06-27',mag_lim=16)

pred
```

```
Obtaining data for Chariklo from SBDB
Ephemeris was split in 1 parts for better search of stars

Searching occultations in part 1/1
Generating Ephemeris between 2017-06-20 00:00:00.000 and 2017-06-26 23:59:00.000 ...
Downloading stars ...
    4 GaiaDR3 stars downloaded
Identifying occultations ...

2 occultations found.
```

[8]:
```
<PredictionTable length=2>
         Epoch             ICRS Star Coord at Epoch    ...  GaiaDR3 Source ID
                                                       ...
         object                      object             ...        str19
----------------------- ------------------------------ ... -------------------
2017-06-21 09:57:43.440 18 55 36.17454 -31 31 19.03261 ... 6760228702284187264
2017-06-22 21:18:48.200 18 55 15.65251 -31 31 21.67062 ... 6760223758801661440
```

[9]:
```
# setting up the Star object
star_occ = Star(code='6760223758801661440')

print(star_occ)
```

```
1 GaiaDR3 star found band={'G': 14.223702}
star coordinate at J2016.0: RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.
↪018 mas

Downloading star parameters from I/297/out
GaiaDR3 star Source ID: 6760223758801661440
ICRS star coordinate at J2016.0:
RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.0180 mas
pmRA=3.556 +/- 0.025 mas/yr, pmDEC=-2.050 +/- 0.020 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.2121 +/- 0.0228 mas, Rad. Vel.=-40.49 +/- 3.73 km/s
```

(continues on next page)

(continued from previous page)

```
Magnitudes: G: 14.224, B: 14.320, V: 13.530, R: 14.180, J: 12.395, H: 11.781,
            K: 11.627

Apparent diameter from Kervella et. al (2004):
    V: 0.0216 mas, B: 0.0216 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0238 mas, V: 0.0244 mas
    ms: B: 0.0261 mas, V: 0.0198 mas
    vs: B: 0.0350 mas, V: 0.0315 mas
```

[10]:
```
# setting up the Occultation object
occ = Occultation(star=star_occ, body=chariklo, time='2017-06-22 21:18')

print(occ)
```

```
Stellar occultation of star GaiaDR3 6760223758801661440 by 10199 Chariklo (1997 CU26).

Geocentric Closest Approach: 0.049 arcsec
Instant of CA: 2017-06-22 21:18:48.200
Position Angle: 359.72 deg
Geocentric shadow velocity: -22.00 km / s
Sun-Geocenter-Target angle:  166.42 deg
Moon-Geocenter-Target angle: 149.11 deg


No observations reported

################################################################################
                                    STAR
################################################################################
GaiaDR3 star Source ID: 6760223758801661440
ICRS star coordinate at J2016.0:
RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.0180 mas
pmRA=3.556 +/- 0.025 mas/yr, pmDEC=-2.050 +/- 0.020 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.2121 +/- 0.0228 mas, Rad. Vel.=-40.49 +/- 3.73 km/s

Magnitudes: G: 14.224, B: 14.320, V: 13.530, R: 14.180, J: 12.395, H: 11.781,
            K: 11.627

Apparent diameter from Kervella et. al (2004):
    V: 0.0216 mas, B: 0.0216 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0238 mas, V: 0.0244 mas
    ms: B: 0.0261 mas, V: 0.0198 mas
    vs: B: 0.0350 mas, V: 0.0315 mas

Geocentric star coordinate at occultation Epoch (2017-06-22 21:18:48.200):
RA=18h55m15.65251s +/- 0.0327 mas, DEC=-31d31m21.6706s +/- 0.0341 mas

################################################################################
                          10199 Chariklo (1997 CU26)
```

(continues on next page)

(continued from previous page)

```
################################################################################
Object Orbital Class: Centaur
Spectral Type:
    SMASS: D  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Relatively featureless spectrum with very steep red slope.
Discovered 1997-Feb-15 by Spacewatch at Kitt Peak

Physical parameters:
Diameter:
    302 +/- 30 km
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),
Rotation:
    7.004 +/- 0 h
    Reference: LCDB (Rev. 2021-June); Warner et al., 2009, [Result based on less than␣
↪full coverage, so that the period may be wrong by 30 percent or so.]  REFERENCE LIST:
↪[Fornasier, S.; Lazzaro, D.; Alvarez-Candal, A.; Snodgrass, C.; et al. (2014) Astron.␣
↪Astrophys. 568, L11.], [Leiva, R.; Sicardy, B.; Camargo, J.I.B.; Desmars, J.; et al.␣
↪(2017) Astron. J. 154, A159.]
Absolute Magnitude:
    6.54 +/- 0 mag
    Reference: MPO691682,
Albedo:
    0.045 +/- 0.01
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),

Ellipsoid: 151.0 x 151.0 x 151.0

----------- Ephemeris -----------

EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

[11]:
```
# defining the observers and their lat, lon, and height information

out = Observer(name='Outeniqua'  ,lon='+16 49 17.710', lat='-21 17 58.170', height =1416)
ond = Observer(name='Onduruquea' ,lon='+15 59 33.750', lat='-21 36 26.040', height =1220)
tiv = Observer(name='Tivoli'     ,lon='+18 01 01.240', lat='-23 27 40.190', height =1344)
whc = Observer(name='Windhoek'   ,lon='+17 06 31.900', lat='-22 41 55.160', height =1902)
hak = Observer(name='Hakos'      ,lon='+16 21 41.320', lat='-23 14 11.040', height =1843)
```

[12]:
```
# Here we are defining each occultation light curve mannualy.
# However, having the data, one can use the fitting methods
# shown above to find the occultation instants and its errors.

out_lc = LightCurve(name='Outeniqua lc',
                    initial_time='2017-06-22 21:20:00.056',
                    end_time ='2017-06-22 21:29:59.963',
```

(continues on next page)

(continued from previous page)

```
                    immersion='2017-06-22 21:21:20.329',immersion_err=0.320,
                    emersion ='2017-06-22 21:21:30.343',emersion_err=0.340)

ond_lc = LightCurve(name='Onduruquea lc',
                    initial_time='2017-06-22 21:11:52.175',
                    end_time ='2017-06-22 21:25:13.389',
                    immersion='2017-06-22 21:21:22.213',immersion_err=0.100,
                    emersion ='2017-06-22 21:21:33.824',emersion_err=0.110)

tiv_lc = LightCurve(name='Tivoli lc',
                    initial_time='2017-06-22 21:16:00.094',
                    end_time ='2017-06-22 21:28:00.018',
                    immersion='2017-06-22 21:21:15.628',immersion_err=0.700,
                    emersion ='2017-06-22 21:21:19.988',emersion_err=0.700)

whc_c14_lc = LightCurve(name='Windhoek C14 lc',
                    initial_time='2017-06-22 21:12:48.250',
                    end_time ='2017-06-22 21:32:47.963',
                    immersion='2017-06-22 21:21:17.609',immersion_err=0.240,
                    emersion ='2017-06-22 21:21:27.564',emersion_err=0.260)

whc_d16_lc = LightCurve(name='Windhoek D16 lc',
                    initial_time='2017-06-22 21:20:01.884',
                    end_time ='2017-06-22 21:22:21.894',
                    immersion='2017-06-22 21:21:17.288',immersion_err=0.280,
                    emersion ='2017-06-22 21:21:27.228',emersion_err=0.340)

hak_lc = LightCurve(name='Hakos lc',
                    initial_time='2017-06-22 21:10:19.461',
                    end_time ='2017-06-22 21:30:19.345')
```

[13]:
```
# Now we will add the defined instants as chords to the occultation object

occ.chords.add_chord(observer=out, lightcurve=out_lc)
occ.chords.add_chord(observer=ond, lightcurve=ond_lc)
occ.chords.add_chord(observer=tiv, lightcurve=tiv_lc)
occ.chords.add_chord(name='Windhoek C14', observer=whc, lightcurve=whc_c14_lc)
occ.chords.add_chord(name='Windhoek D16', observer=whc, lightcurve=whc_d16_lc)
occ.chords.add_chord(observer=hak, lightcurve=hak_lc)


# And fix some of the time offsets measured for each light curve.
# (This information is provided by the observers, usually due to GPS time precision)
out_lc.dt = -0.150
ond_lc.dt = -0.190
tiv_lc.dt = -0.150
whc_c14_lc.dt = -0.375
whc_d16_lc.dt = +0.000
```

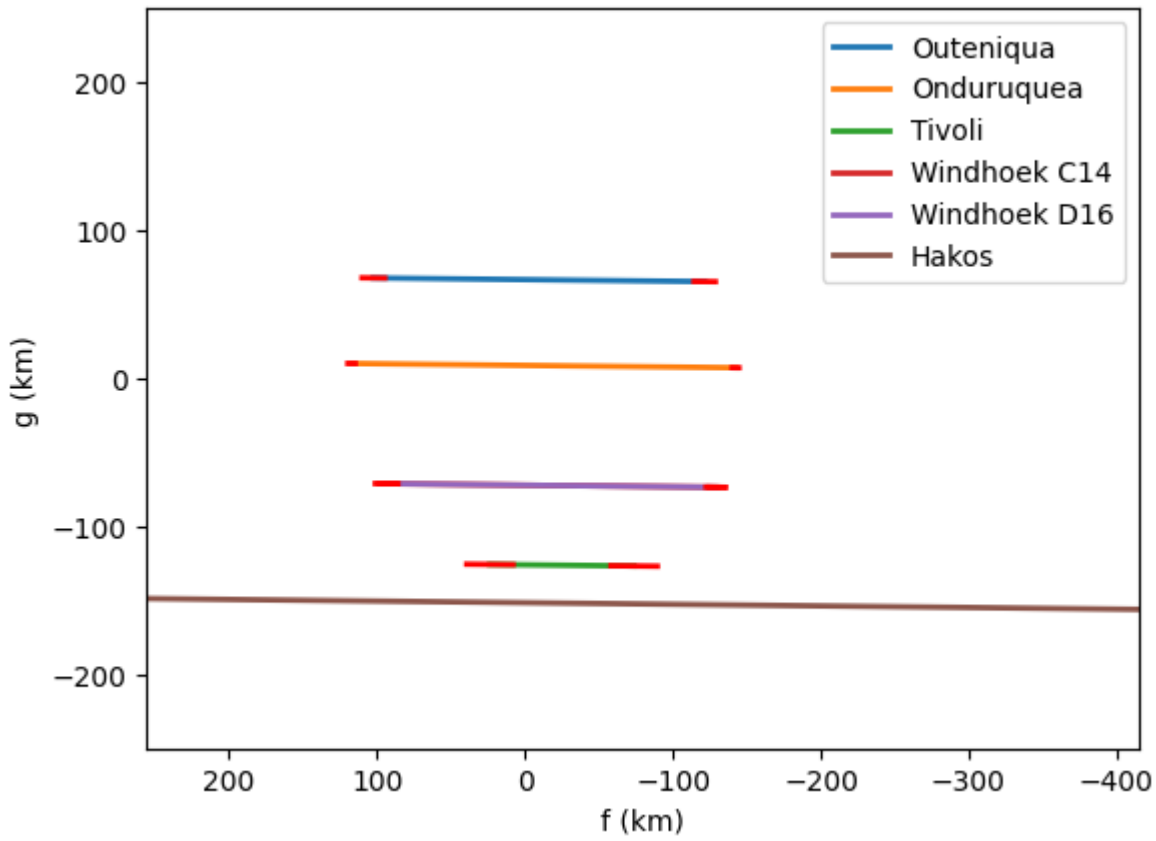
```
/mnt/data/repos/SORA/sora/body/core.py:332: UserWarning: H and/or G is not defined for␣
↪10199 Chariklo. Searching into JPL Horizons service
  warnings.warn('H and/or G is not defined for {}. Searching into JPL Horizons service'.
```

(continues on next page)

(continued from previous page)

```
↪format(self.shortname))
```

[14]:
```
# plotting the added chords
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```



To fit an ellipse to the data, as before, we will set a small value for the "number_chi" variable (which plays a role similar to "loop" in the light curve fit) since we first want only a best-fit value obtained with the convergence methods.

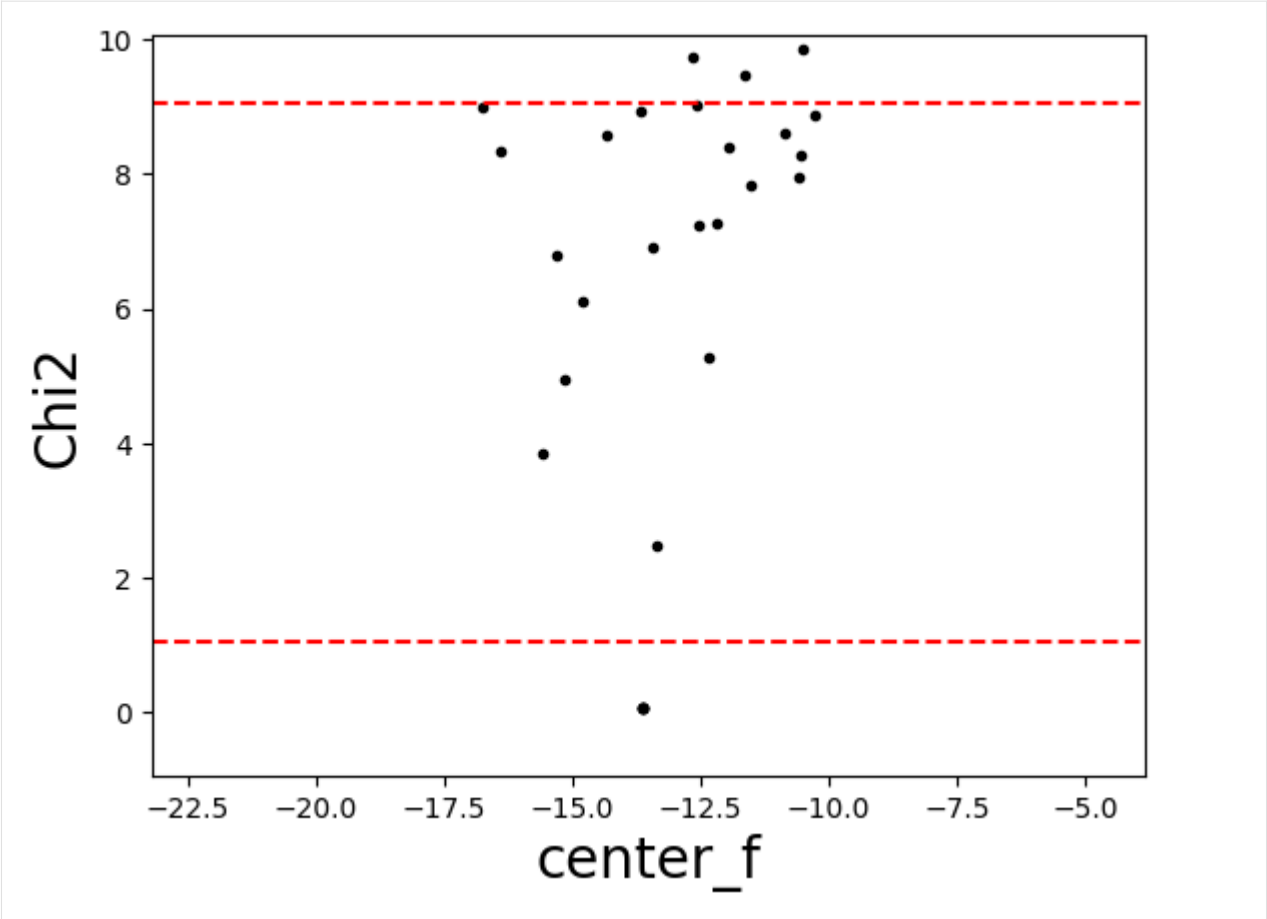
[15]:
```
### If this still takes to long to run, reduce the number_chi value

ellipse_chi2  = occ.fit_ellipse(center_f=0, center_g=0, dcenter_f=500, dcenter_g=500,
                                equatorial_radius=200, dequatorial_radius=200,␣
↪oblateness=0.0,
                                doblateness=0.2, position_angle=90, dposition_angle=90,
                                dchi_min=10, number_chi=2500, method='ls', threads=5,␣
↪verbose=True)
```

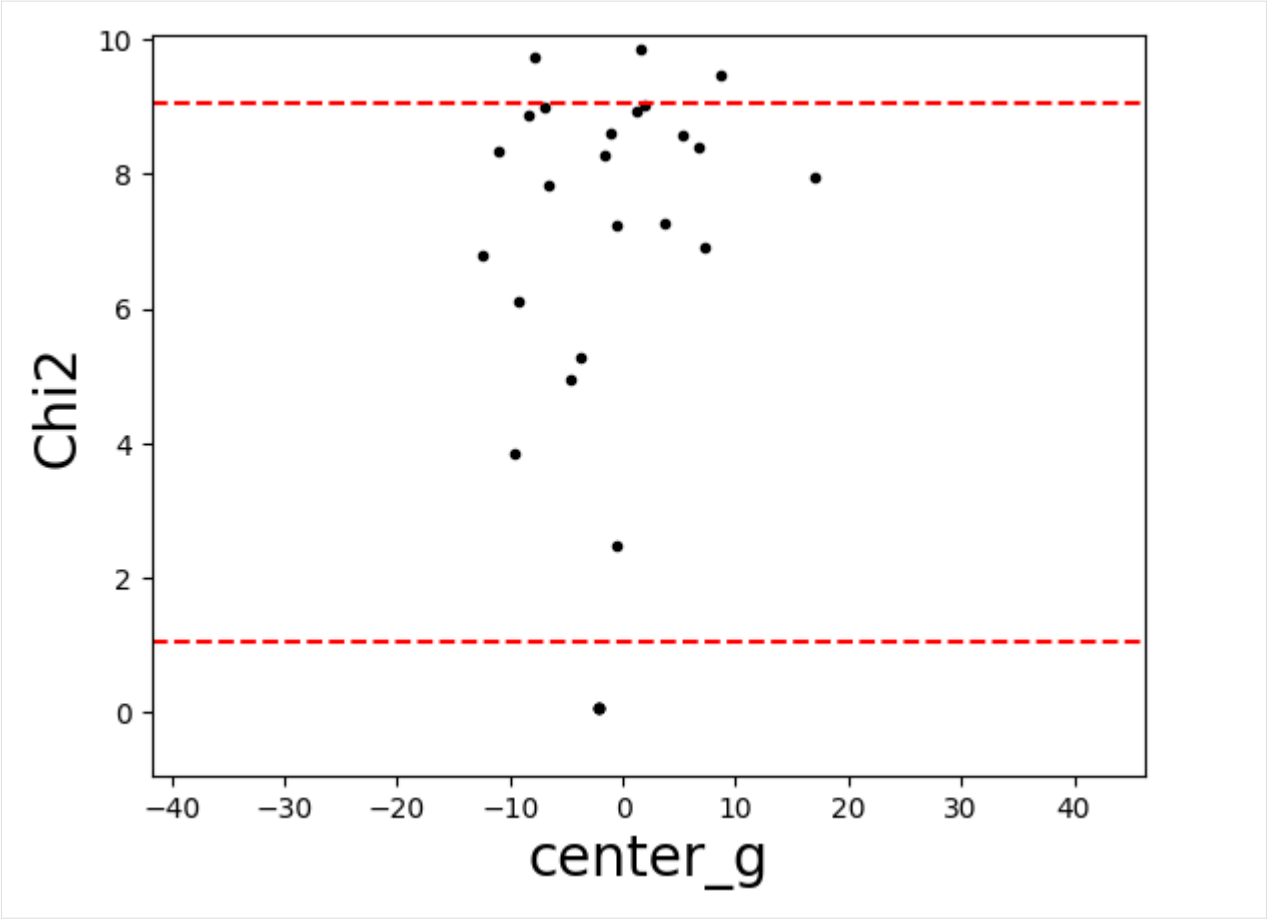
(continues on next page)

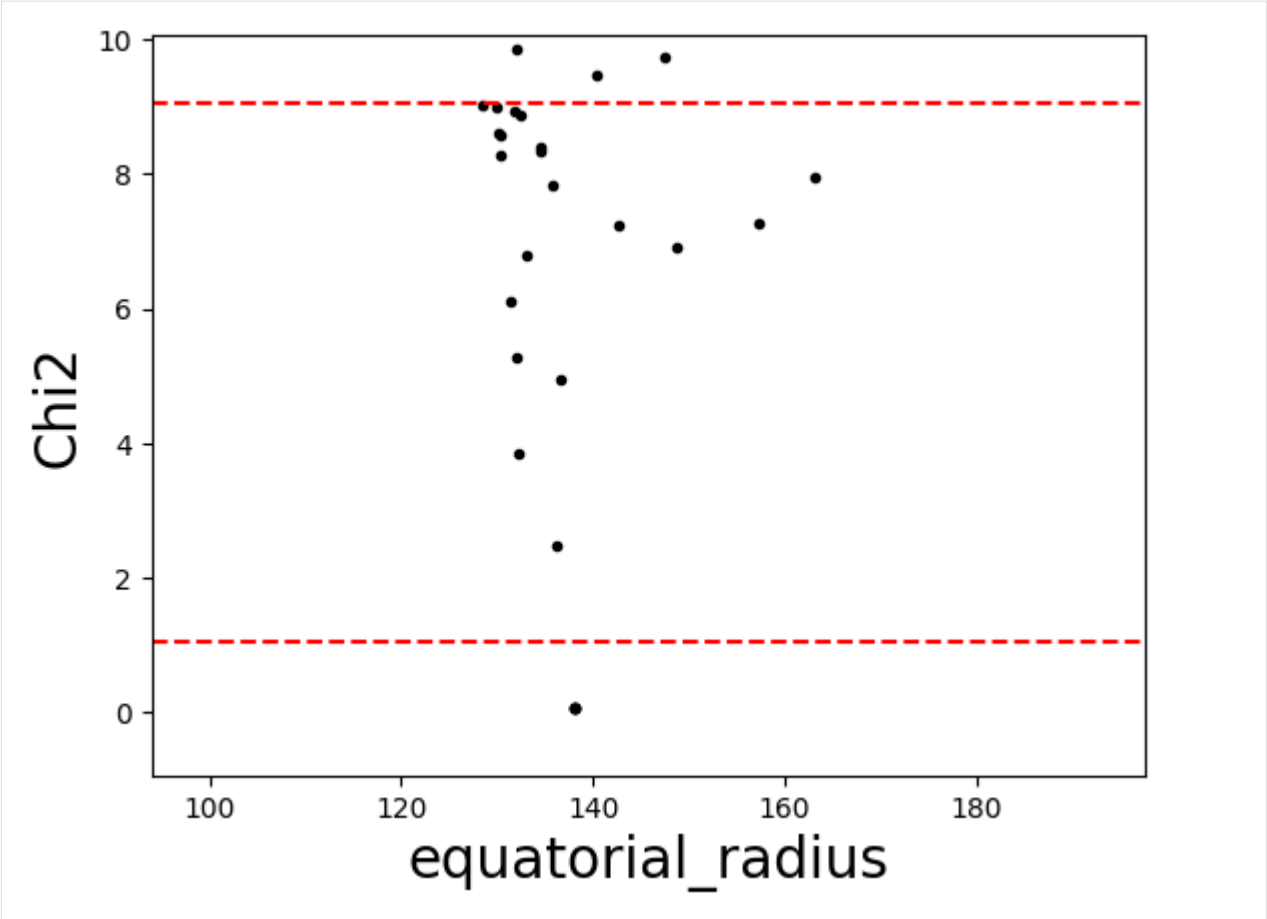
(continued from previous page)

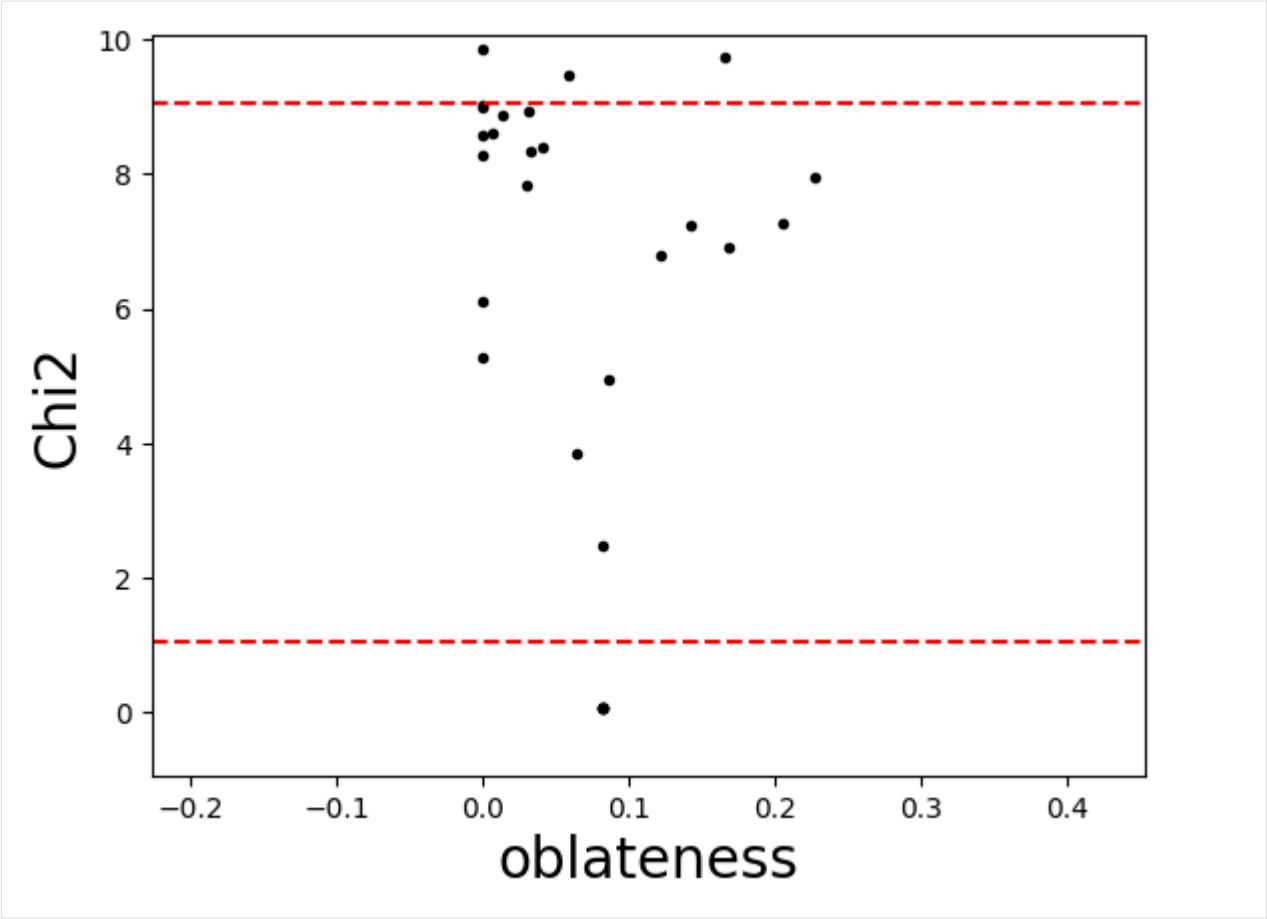
```
# print the best fit variables
print(ellipse_chi2)

# print the chi-square maps
ellipse_chi2.plot_chi2()
```

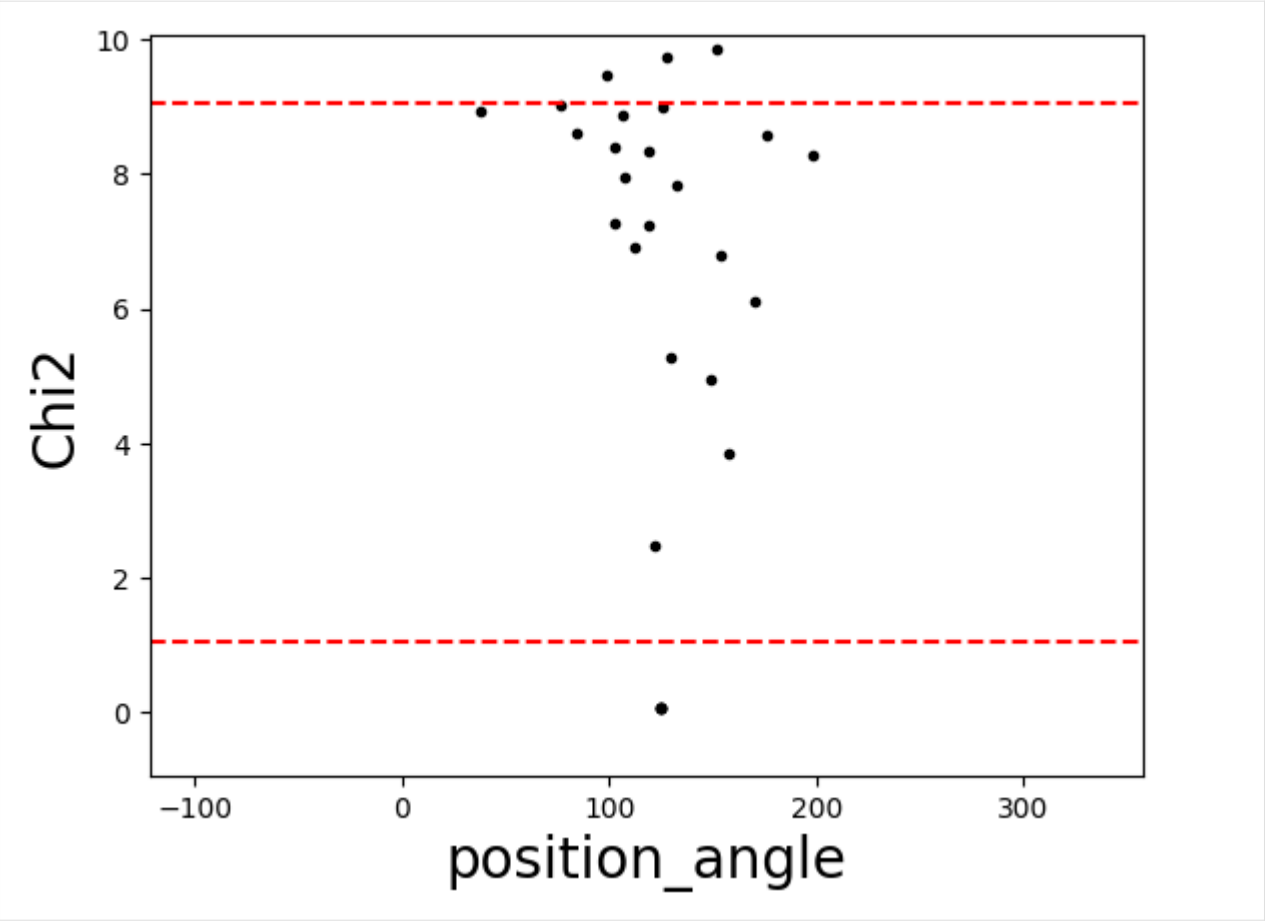
```
Ellipse fit: ||  - 100%
Total elapsed time: 7.595 seconds.
Minimum chi-square: 0.062
Number of fitted points: 10
Number of fitted parameters: 5
Minimum chi-square per degree of freedom: 0.012

center_f:
    1-sigma: -13.624 +/- 0.000
    3-sigma: -13.515 +/- 3.227

center_g:
    1-sigma: -2.240 +/- 0.000
    3-sigma: 2.264 +/- 14.672

equatorial_radius:
    1-sigma: 138.043 +/- 0.000
    3-sigma: 145.778 +/- 17.264

oblateness:
    1-sigma: 0.082 +/- 0.000
    3-sigma: 0.113 +/- 0.113

position_angle:
    1-sigma: 125.073 +/- 0.000
    3-sigma: 117.883 +/- 79.834
```


Chi2
10
8
6
4
2
0
−22.5
−20.0
−17.5
−15.0
−12.5
−10.0
−7.5
−5.0
center_f


Chi2
center_g
10
8
6
4
2
0
−40
−30
−20
−10
0
10
20
30
40


10
8
6
4
2
0
Chi2
100
120
140
160
180
equatorial_radius


Chi2
oblateness
10
8
6
4
2
0
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4



Knowing better the region of the minima for the parameters and having a coarse approximation their the 3-sigma boundaries, we can now set a larger value for the "number_chi" variable and explore more efficiently the space parameter.

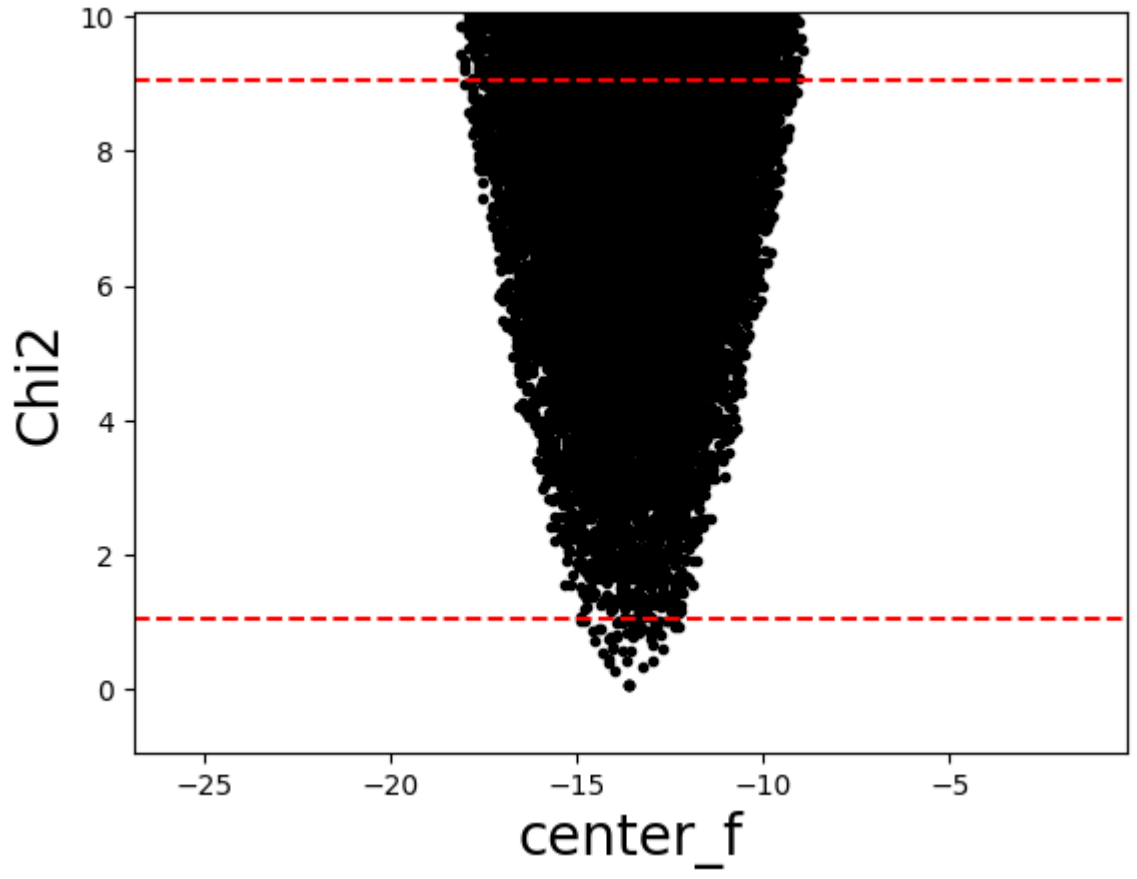
[16]:
```
ellipse_chi2  = occ.fit_ellipse(center_f=-13.6, center_g=-2.2, dcenter_f=25, dcenter_
↪g=25,
                                equatorial_radius=138, dequatorial_radius=50,␣
↪oblateness=0.08,
                                doblateness=0.2, position_angle=125, dposition_angle=90 ,
                                dchi_min=10, number_chi=50000, method='ls', threads=5,␣
↪verbose=True)


# print the best fit variables
print(ellipse_chi2)

# print the chi-square maps
ellipse_chi2.plot_chi2()
```

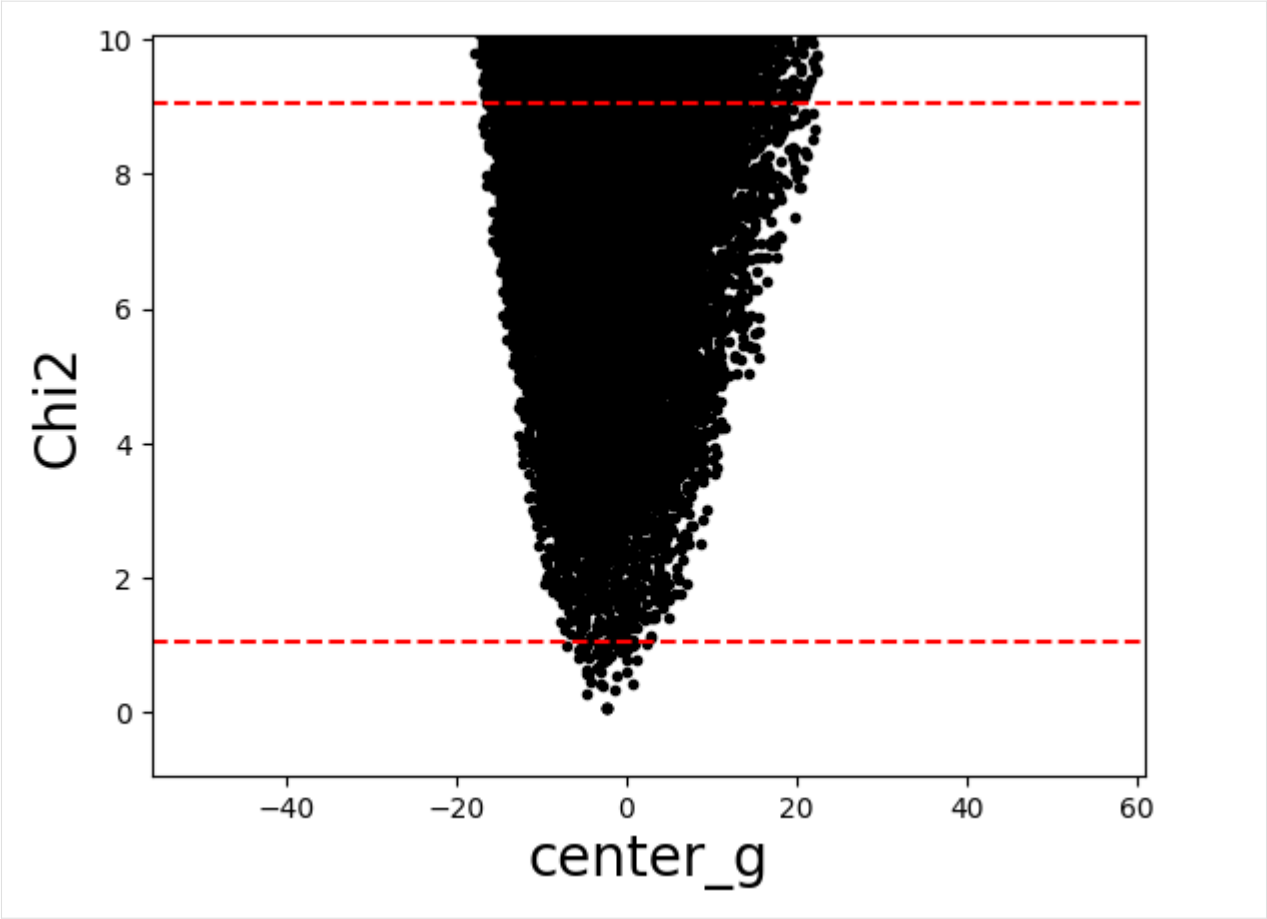
```
Ellipse fit: ||  - 100%
Total elapsed time: 5.022 seconds.
Minimum chi-square: 0.062
Number of fitted points: 10
Number of fitted parameters: 5
```

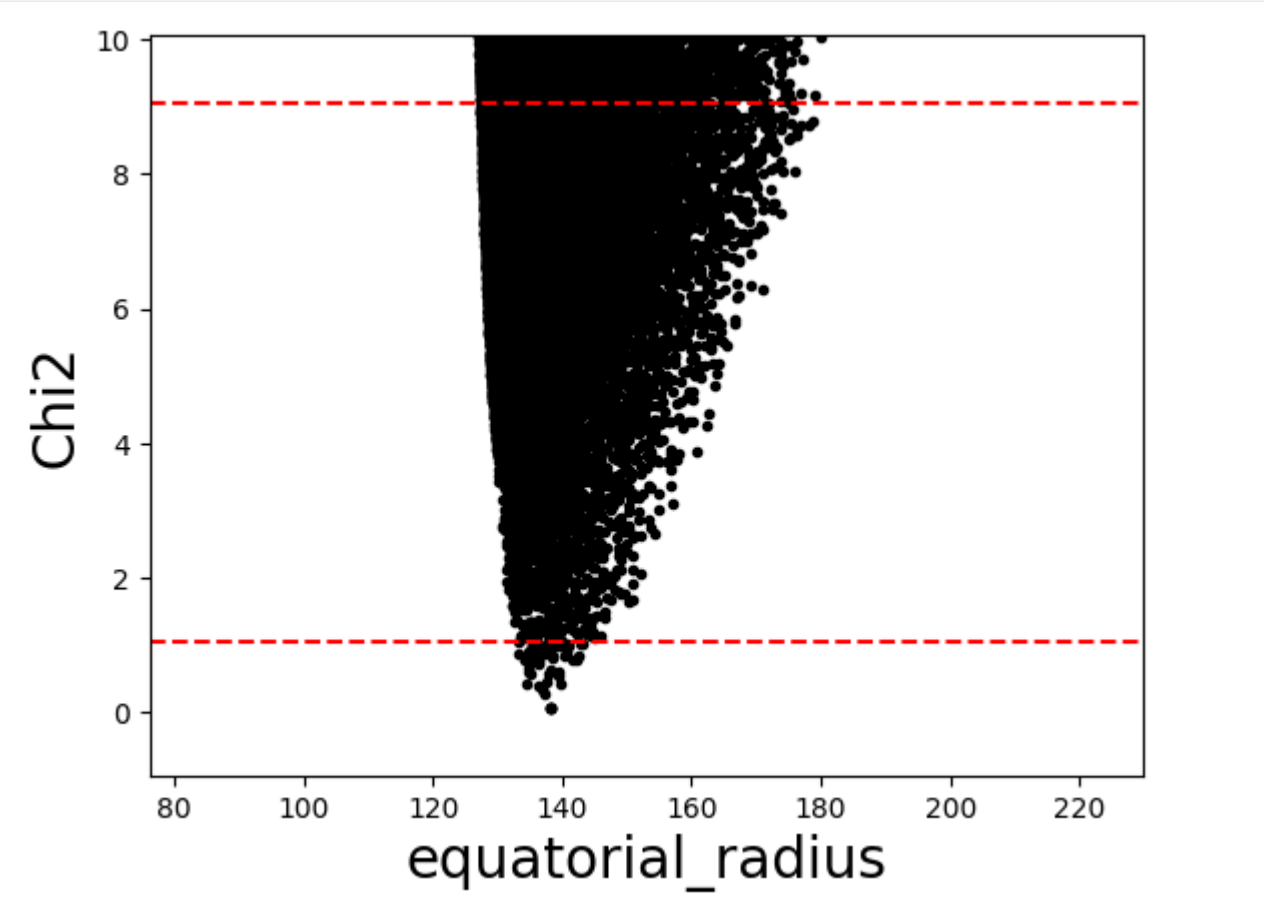
(continues on next page)

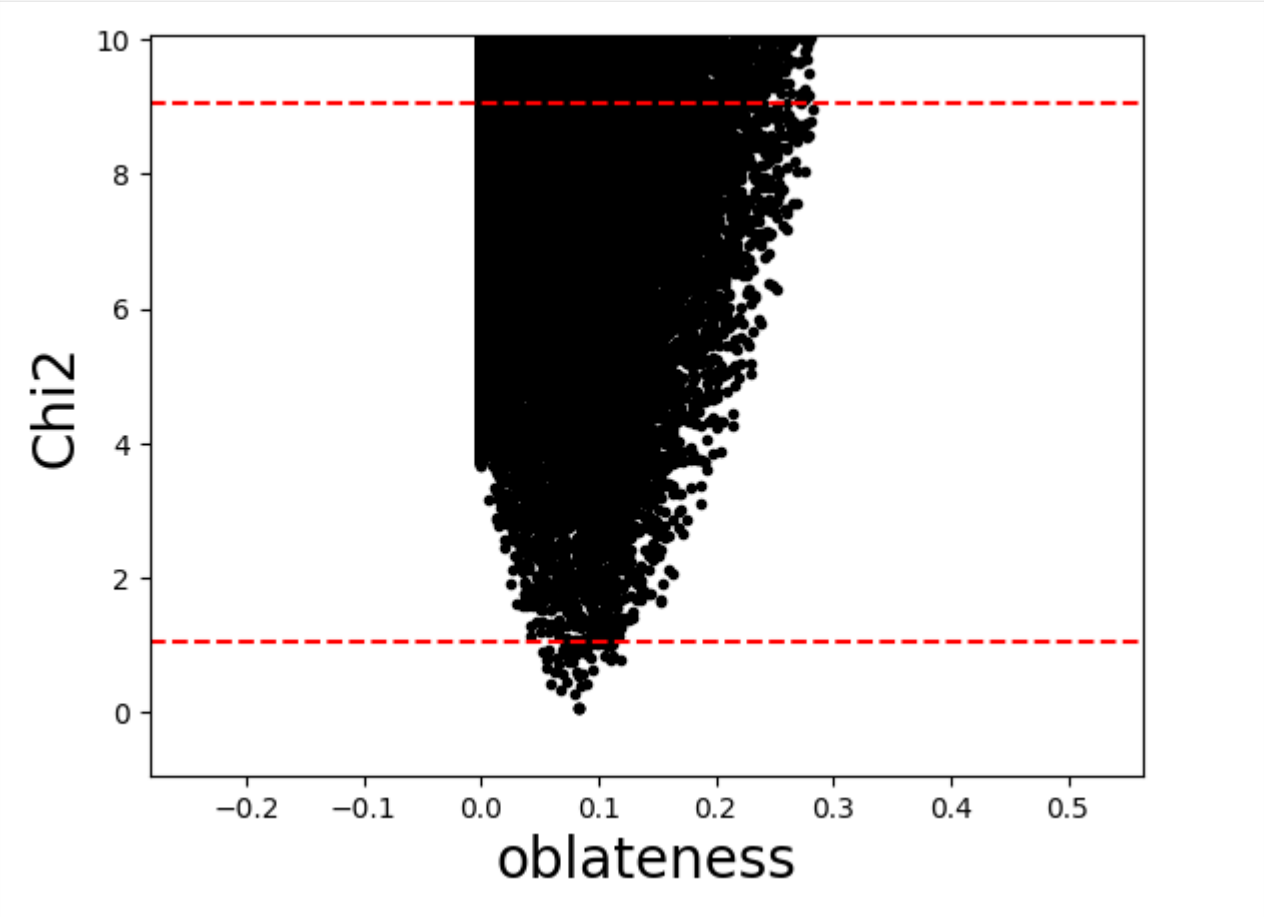
(continued from previous page)

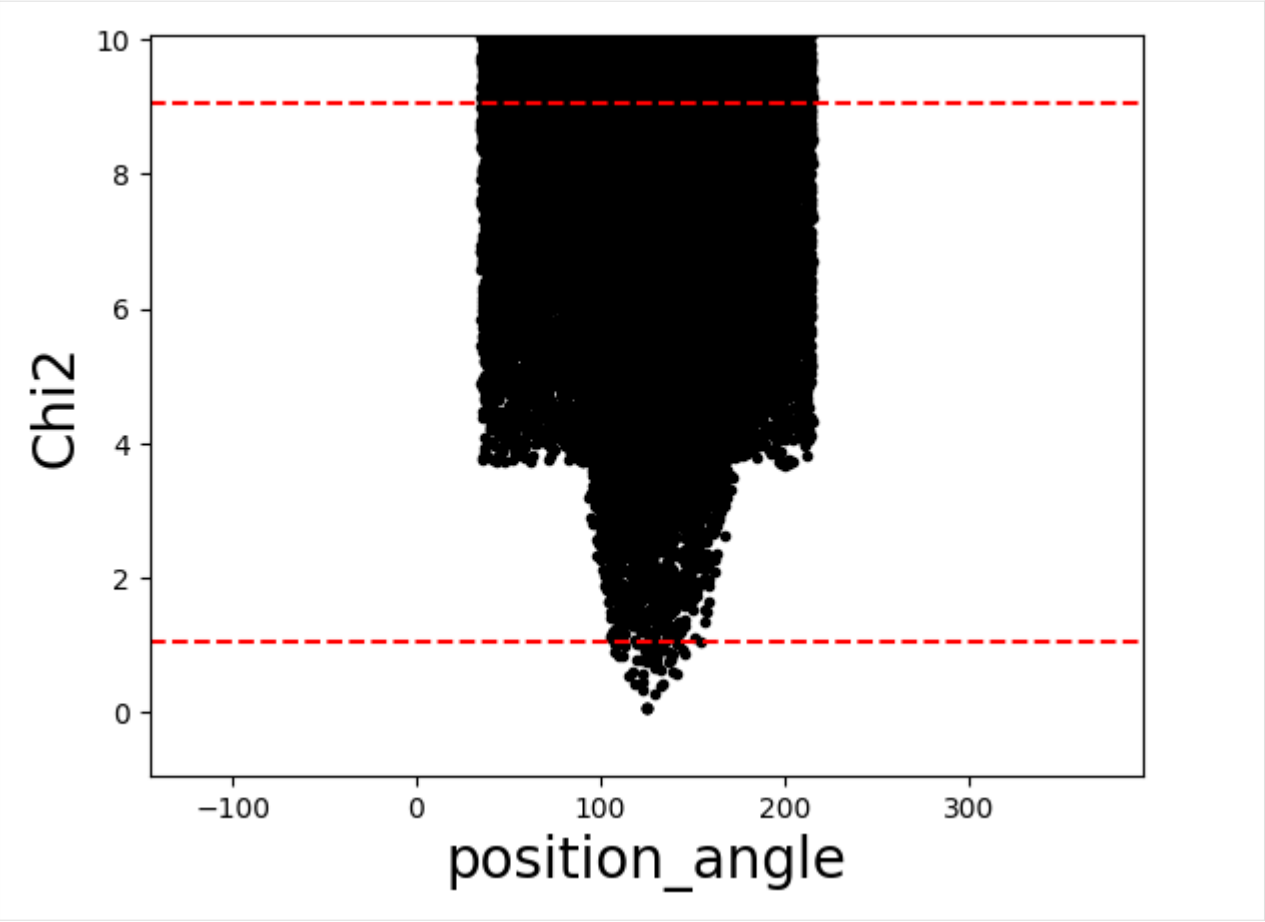
```
Minimum chi-square per degree of freedom: 0.012

center_f:
    1-sigma: -13.617 +/- 1.317
    3-sigma: -13.572 +/- 4.442

center_g:
    1-sigma: -2.349 +/- 4.639
    3-sigma: 2.609 +/- 19.447

equatorial_radius:
    1-sigma: 138.215 +/- 4.864
    3-sigma: 153.012 +/- 25.587

oblateness:
    1-sigma: 0.085 +/- 0.033
    3-sigma: 0.141 +/- 0.141

position_angle:
    1-sigma: 131.015 +/- 23.334
    3-sigma: 125.085 +/- 89.974
```




Chi2
center_g
10
8
6
4
2
0
−40
−20
0
20
40
60

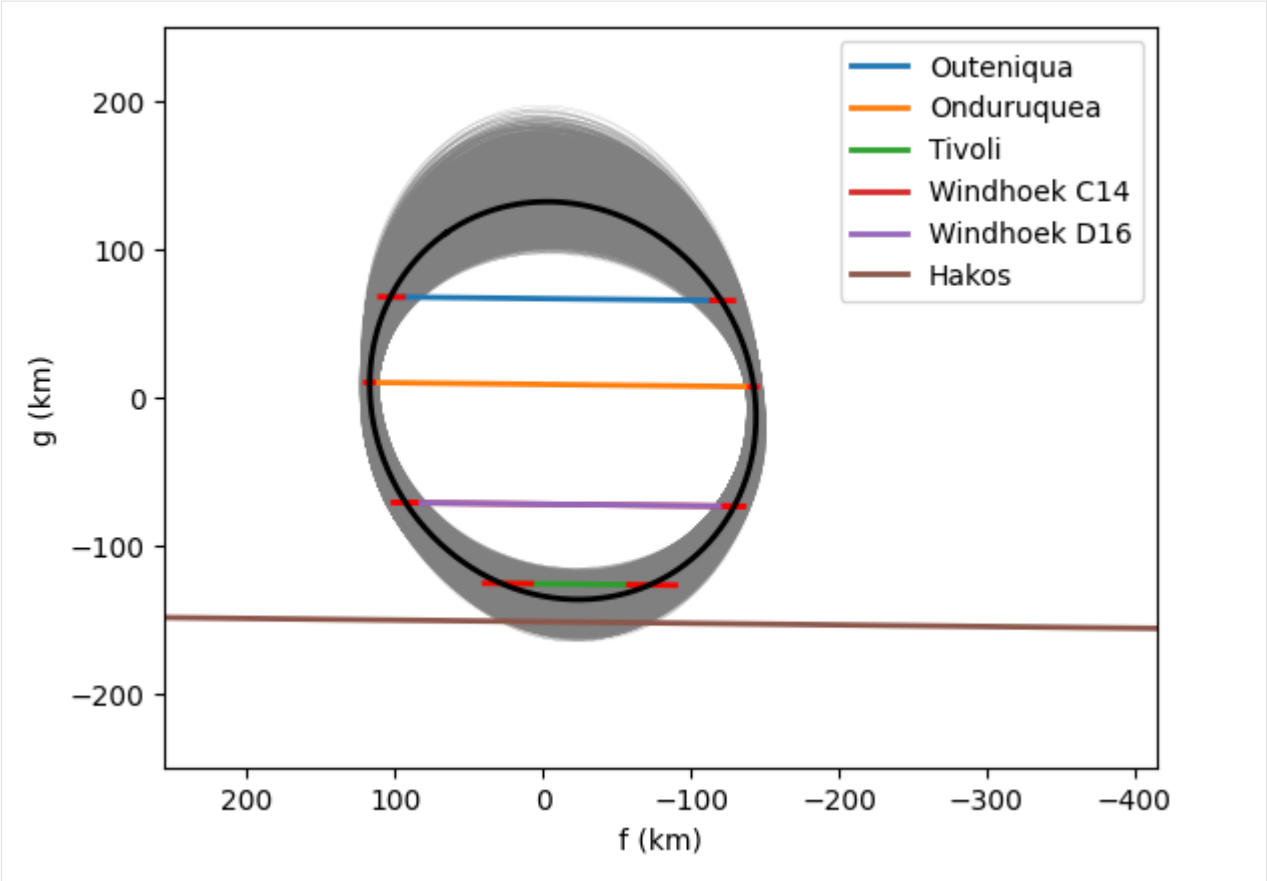

Chi2
10
8
6
4
2
0
80
100
120
140
160
180
200
220
equatorial_radius


Chi2
oblateness
10
8
6
4
2
0
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5



Let's finally plot our solution and the region of uncertainties around this best set of parameters.

[17]:
```
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

#plotting the best fitted ellipse, in black
draw_ellipse(**ellipse_chi2.get_values())

# ploting all the ellipses within 3-sigma, in gray
draw_ellipse(**ellipse_chi2.get_values(sigma=3),alpha=1.0)

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```


Outeniqua
Onduruquea
Tivoli
Windhoek C14
Windhoek D16
Hakos
200
100
0
−100
−200
g (km)
200
100
0
−100
−200
−300
−400
f (km)

CHAPTER

NINE

JUPYTER-NOTEBOOK GUIDELINES

All the Jupyter-Notebooks can be downloaded `here`

9.1 Body Class

The `Body` Class within SORA was created to handle the information about the Solar System object. The online documentation here contains the details about every step.

This Jupyter-Notebook was designed as a tutorial for how to work with the `Body` Class. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The ``Body`` Docstring was designed to help the users. Also, each function has its Docstring containing its main purpose and the needed parameters (physical description and formats). Please, do not hesitate to use it.

9.1.1 0. Index

[1]:
```
## import the Body Class
from sora.body import Body

## To facilitate, sora allows to import Body directly from the root
from sora import Body
```

```
SORA version: 0.3
```

9.1.2 1. Instantiating a Body object

A Solar System Body have a lot of parameters that characterizes itself. The SORA Body Class is able to download these parameter from the JPL Small Body DataBase. For that, a name should be given to search the object.

[2]:
```
Body?
```

```
Init signature: Body(name, database='auto', **kwargs)
Docstring:
Class that contains and manages the information of the body.

Attributes
----------
name : `str`, required
    The name of the object. It can be the used `spkid` or `designation
    number` to query the SBDB (Small-Body DataBase). In this case, the name
    is case insensitive.

database : `str`, optional, default='auto'
    The database to query the object. It can be ``satdb`` for our temporary
    hardcoded satellite database, or ```'sbdb'``` to query on the SBDB. If
    database is set as ``auto`` it will try first with ``satdb``,
    then ``sbdb``. If the user wants to use their own information,
    database must be given as ``None``. In this case, `spkid` parameter
    must be given.

ephem : `sora.EphemKernel`, `sora.EphemHorizons`, `sora.EphemJPL`, `sora.EphemPlanete`
    An Ephem Class that contains information about the ephemeris. It can be
    "horizons" to automatically defined an EphemHorizons object or a list of
    kernels to automatically define an EphemKernel object.

orbit_class : `str`
    It defines the Orbital class of the body. It can be ``TNO``,
    ``Satellite``, ``Centaur``, ``comet``, ``asteroid``, ``trojan``, ``neo``,
    and ``planet``. It is important for a better characterization of the
    object. If a different value is given, it will be defined as
    ``unclassified``.

spkid : `str`, `int`, `float`
    If ``database=None``, the user must give a `spkid` or an `ephem`
    which has the `spkid` parameter.

shape : `str`, `sora.body.shape.Shape3D`
    It defines the input shape of the body. It can be a body.shape object
    or the path to OBJ file.

albedo : `float`, `int`
    The albedo of the object.

H : `float`, `int`
    The absolute magnitude.

G : `float`, `int`
    The phase slope.
```

(continues on next page)

(continued from previous page)

```
diameter : `float`, `int`, `astropy.quantity.Quantity`
    The diameter of the object, in km.

density : `float`, `int`, `astropy.quantity.Quantity`
    The density of the object, in g/cm³.

GM : `float`, `int`, `astropy.quantity.Quantity`
    The Standard Gravitational Parameter, in km³/s².

rotation : `float`, `int`, `astropy.quantity.Quantity`
    The Rotation of the object, in hours.

pole : `str`, `astropy.coordinates.SkyCoord`
    The Pole coordinates of the object. It can be a `SkyCoord object` or a
    string in the format ``'hh mm ss.ss +dd mm ss.ss'``.

BV : `float`, `int`
    The B-V color.

UB : `float`, `int`
    The U-B color.

smass : `str`
    The spectral type in SMASS classification.

tholen : `str`
    The spectral type in Tholen classification.

Note
----
The following attributes are are returned from the Small-Body DataBase when
``database='sbdb'`` or from our temporary hardcoded Satellite DataBase when
``database='satdb'``:

`orbit_class`, `spkid`, `albedo`, `H`, `G`, `diameter`, `density`, `GM`,
`rotation`, `pole`, `BV`, `UB`, `smass`, and `tholen`.

These are physical parameters the user can give to the object. If a query is
made and user gives a parameter, the parameter given by the user is defined
in the *Body* object.
File:           ~/Documentos/códigos/SORA/sora/body/core.py
Type:           type
Subclasses:
```

[3]:
```
ceres = Body(name='Ceres')
```

```
Obtaining data for Ceres from SBDB
```

[4]:
```
print(ceres)
```

```
###############################################################################
                                1 Ceres (A801 AA)
###############################################################################
Object Orbital Class: Main-belt Asteroid
Spectral Type:
    SMASS: C  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
    Tholen: G [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Linear, generally featureless spectra. Differences in UV absorption features and␣
↪presence/absence of narrow absorption feature near 0.7 m.
Discovered 1801-Jan-01 by Piazzi, G. at Palermo

Physical parameters:
Diameter:
    939.4 +/- 0.2 km
    Reference: Nature vol. 537, pp515-517 (22 September 2016),
Mass:
    9.3835e+20 +/- 1.3485e+16 kg
    Reference: Nature vol. 537, pp515-517 (22 September 2016),
Density:
    2.162 +/- 0.008 g / cm3
    Reference: Nature vol. 537, pp515-517 (22 September 2016),
Rotation:
    9.0742 +/- 1e-06 h
    Reference: Nature vol. 537, pp515-517 (22 September 2016), derived from published␣
↪value: 952.1532 +/- 0.0001 deg./day
Pole
    RA:291d25m04.8s +/- 0h00m00s
    DEC:66d45m50.4s +/- 0d00m00s
    Reference: User,
Absolute Magnitude:
    3.33 +/- 0 mag
    Reference: MPO691494, IRAS observations used: 15
Phase Slope:
    0.12 +/- 0
    Reference: PDS3 (MPC 17257), Fit
Albedo:
    0.09 +/- 0.003
    Reference: Li et al. (2006) Icarus v182:pp143-160, V-band geometric albedo
B-V color:
    0.713 +/- 0.014
    Reference: EAR-A-5-DDR-UBV-MEAN-VALUES-V1.2, #obs=52; phase (min.=1.47, mean=11.85,␣
↪max.=22.59) deg.
U-B color:
    0.426 +/- 0.026
    Reference: EAR-A-5-DDR-UBV-MEAN-VALUES-V1.2, #obs=52; phase (min.=1.47, mean=12.16,␣
↪max.=22.59) deg.

PlanetocentricFrame:
    Epoch: J2000.000
    alpha_pole = 291.418 +0.000000*T +0.000000
    delta_pole = 66.764 +0.000000*T +0.000000
    W = 170.65 +952.153200*d +0.000000
    Reference: Archinal, B. A., Report of the IAU Working Group on Cartographic␣
```

(continues on next page)

(continued from previous page)

```
↪Coordinates and Rotational Elements: 2015, Celestial Mechanics and Dynamical Astronomy,
↪ (2018) 130:22

Ellipsoid: 482.2 x 482.1 x 445.9

----------- Ephemeris -----------

EphemHorizons: Ephemeris are downloaded from Horizons website (SPKID=2000001)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

Other parameters that are interpreted by the Body Class are downloaded from the SBDB. They are accessed vi the attribute Body.meta_sbdb

[5]:
```
ceres.meta_sbdb
```

[5]:
```
OrderedDict([('phys_par',
              OrderedDict([('H', 3.33),
                           ('H_sig', None),
                           ('H_ref', 'MPO691494'),
                           ('H_note', 'IRAS observations used: 15'),
                           ('G', 0.12),
                           ('G_sig', None),
                           ('G_ref', 'PDS3 (MPC 17257)'),
                           ('G_note', 'Fit'),
                           ('diameter', <Quantity 939.4 km>),
                           ('diameter_sig', <Quantity 0.2 km>),
                           ('diameter_ref',
                            'Nature vol. 537, pp515-517 (22 September 2016)'),
                           ('diameter_note', None),
                           ('extent', '964.4 x 964.2 x 891.8'),
                           ('extent_sig', '0.2 x 0.2 x 0.2'),
                           ('extent_ref',
                            'Nature vol. 537, pp515-517 (22 September 2016)'),
                           ('extent_note',
                            'fit w.r.t. center of mass and in body-fixed frame'),
                           ('GM', <Quantity 62.6284 km3 / s2>),
                           ('GM_sig', <Quantity 0.0009 km3 / s2>),
                           ('GM_ref',
                            'Nature vol. 537, pp515-517 (22 September 2016)'),
                           ('GM_note', None),
                           ('density', <Quantity 2.162 g / cm3>),
                           ('density_sig', <Quantity 0.008 g / cm3>),
                           ('density_ref',
                            'Nature vol. 537, pp515-517 (22 September 2016)'),
                           ('density_note', None),
                           ('rot_per', <Quantity 9.07417 h>),
                           ('rot_per_sig', <Quantity 1.e-06 h>),
                           ('rot_per_ref',
                            'Nature vol. 537, pp515-517 (22 September 2016)'),
                           ('rot_per_note',
```

(continues on next page)

(continued from previous page)

```
                          'derived from published value: 952.1532 +/- 0.0001 deg./day
↪'),
                         ('pole', '291.421/66.758'),
                         ('pole_sig', '0.007/0.002'),
                         ('pole_ref',
                          'Nature vol. 537, pp515-517 (22 September 2016)'),
                         ('pole_note', None),
                         ('albedo', 0.09),
                         ('albedo_sig', 0.003),
                         ('albedo_ref',
                          'Li et al. (2006) Icarus v182:pp143-160'),
                         ('albedo_note', 'V-band geometric albedo'),
                         ('BV', 0.713),
                         ('BV_sig', 0.014),
                         ('BV_ref', 'EAR-A-5-DDR-UBV-MEAN-VALUES-V1.2'),
                         ('BV_note',
                          '#obs=52; phase (min.=1.47, mean=11.85, max.=22.59) deg.'),
                         ('UB', 0.426),
                         ('UB_sig', 0.026),
                         ('UB_ref', 'EAR-A-5-DDR-UBV-MEAN-VALUES-V1.2'),
                         ('UB_note',
                          '#obs=52; phase (min.=1.47, mean=12.16, max.=22.59) deg.'),
                         ('spec_T', 'G'),
                         ('spec_T_sig', 'None'),
                         ('spec_T_ref', 'EAR-A-5-DDR-TAXONOMY-V4.0'),
                         ('spec_T_note',
                          'used 7 color indices; used ground-based radiometric albedo
↪'),
                         ('spec_B', 'C'),
                         ('spec_B_sig', 'None'),
                         ('spec_B_ref', 'EAR-A-5-DDR-TAXONOMY-V4.0'),
                         ('spec_B_note',
                          'based on a high-resolution spectrum by Xu et al. (1995) or␣
↪Bus and Binzel (2002)')])),
             ('object',
              OrderedDict([('shortname', '1 Ceres'),
                         ('neo', False),
                         ('orbit_class',
                          OrderedDict([('name', 'Main-belt Asteroid'),
                                       ('code', 'MBA')])),
                         ('pha', False),
                         ('spkid', '2000001'),
                         ('kind', 'an'),
                         ('orbit_id', '48'),
                         ('fullname', '1 Ceres (A801 AA)'),
                         ('des', '1'),
                         ('prefix', None)])),
             ('signature',
              OrderedDict([('source',
                          'NASA/JPL Small-Body Database (SBDB) API'),
                         ('version', '1.3')])),
             ('orbit',
```

(continues on next page)

(continued from previous page)

```
             OrderedDict([('source', 'JPL'),
                          ('cov_epoch', <Quantity 2458849.5 d>),
                          ('moid_jup', <Quantity 2.09403 AU>),
                          ('t_jup', '3.310'),
                          ('condition_code', '0'),
                          ('not_valid_before', None),
                          ('rms', '.43153'),
                          ('model_pars', []),
                          ('orbit_id', '48'),
                          ('producer', 'Davide Farnocchia'),
                          ('first_obs', '1995-01-05'),
                          ('soln_date', '2021-04-13 11:04:44'),
                          ('two_body', None),
                          ('epoch', <Quantity 2458849.5 d>),
                          ('elements',
                           OrderedDict([('e', 0.07687465013145245),
                                        ('e_sig', 4.82457301562052e-12),
                                        ('q', <Quantity 2.55640115 AU>),
                                        ('q_sig',
                                         <Quantity 1.96580406e-11 AU>),
                                        ('tp', <Quantity 2458240.17913094 d>),
                                        ('tp_sig',
                                         <Quantity 3.72607486e-08 d>),
                                        ('om', <Quantity 80.30119019 deg>),
                                        ('om_sig',
                                         <Quantity 6.17186303e-08 deg>),
                                        ('w', <Quantity 73.80896809 deg>),
                                        ('w_sig',
                                         <Quantity 6.61818138e-08 deg>),
                                        ('i', <Quantity 10.59127767 deg>),
                                        ('i_sig',
                                         <Quantity 4.61328999e-09 deg>)])),
                          ('equinox', 'J2000'),
                          ('data_arc', '9520'),
                          ('not_valid_after', None),
                          ('n_del_obs_used', '60'),
                          ('sb_used', 'SB441-N373'),
                          ('n_obs_used', '1075'),
                          ('comment', None),
                          ('pe_used', 'DE441'),
                          ('last_obs', '2021-01-28'),
                          ('moid', <Quantity 1.59232 AU>),
                          ('n_dop_obs_used', '0')])),
            ('discovery',
             OrderedDict([('cref', None),
                          ('location', 'Palermo'),
                          ('ref', 'DISCOVERY.DB'),
                          ('date', '1801-Jan-01'),
                          ('name', None),
                          ('discovery',
                           'Discovered 1801-01-01 by Piazzi, G. at Palermo'),
                          ('site', None),
```

(continues on next page)

(continued from previous page)

```
                            ('who', 'Piazzi, G.'),
                            ('citation', None)]))])
```

The SORA team also created a hardcoded database with planetary satellites and planets to overcome the missing of these data on the SBDB. It is an interim solution and will be removed when a more reliable database is implemented.

[6]:
```
himalia = Body(name='Himalia', database='satdb')
print(himalia)
```

```
################################################################################
                                    Himalia
################################################################################
Object Orbital Class: Natural Satellite


Physical parameters:
Diameter:
    139.6 +/- 1.7 km
    Reference: Grav et al. (2015). AJ 809(1):3,
Mass:
    4.1952e+18 +/- 5.9931e+17 kg
    Reference: Emelyanov (2005). A&A 438(3):L33-L36,
Rotation:
    7.7819 +/- 0.0005 h
    Reference: Pilcher et al. (2012). Icarus 219(2):741-742,
Absolute Magnitude:
    8 +/- 0.01 mag
    Reference: Rettig et al. (2001). Icarus 154(2):313-320,
Phase Slope:
    0.1 +/- 0.15
    Reference: ,
Albedo:
    0.057 +/- 0.008
    Reference: Grav et al. (2015). AJ 809(1):3,

Ellipsoid: 69.8 x 69.8 x 69.8

----------- Ephemeris -----------

EphemHorizons: Ephemeris are downloaded from Horizons website (SPKID=506)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

Finally, the user can also define a `Body` object without any database. In this case, it is required to define the parameters: `[name, spkid, database=None, orbit_class]`

[7]:
```
elara = Body(name='Elara', database=None, spkid=507, orbit_class='satellite')
print(elara)
```

```
################################################################################
```

(continues on next page)

(continued from previous page)

```
                                        Elara
################################################################################
Object Orbital Class: Natural Satellite


Physical parameters:

----------- Ephemeris -----------

EphemHorizons: Ephemeris are downloaded from Horizons website (SPKID=507)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

9.1.3 2. Body Physical Attributes

The parameters shown in the print can be accessed by its attribute name.

These paremeter also have uncertainty and reference, which may be edited

[8]:
```
ceres.diameter
```

[8]: 939.4 km

[9]:
```
ceres.diameter.uncertainty
```

[9]: 0.2 km

[10]:
```
ceres.diameter.reference
```

[10]: 'Nature vol. 537, pp515-517 (22 September 2016)'

[11]:
```
print(ceres.diameter)
```

```
Diameter:
    939.4 +/- 0.2 km
    Reference: Nature vol. 537, pp515-517 (22 September 2016),
```

2.1. Editing Body Attributes

To edit a parameter, simply redefine the attribute. If the core value is edited, the uncertainty is redefined to "0", and the reference to "user"

[12]:
```
ceres.diameter = 972
```

[13]:
```
print(ceres.diameter)
```

```
Diameter:
    972 +/- 0 km
    Reference: User,
```

To correct the uncertainty and reference, just redefine them

[14]:
```
ceres.diameter.uncertainty = 6
ceres.diameter.reference = "Gomes-Júnior et al. (2015), MNRAS 451(3), 2295-2302"
ceres.diameter.notes = "Equatorial Diameter"
```

[15]:
```
print(ceres.diameter)
```

```
Diameter:
    972 +/- 6 km
    Reference: Gomes-Júnior et al. (2015), MNRAS 451(3), 2295-2302, Equatorial Diameter
```

Finally, if you want to edit all the values of a single parameter directly, you must use the PhysicalData Class located in sora.body

[16]:
```
from sora.body import PhysicalData
import astropy.units as u
PhysicalData?
```

```
Init signature:
PhysicalData(
    name,
    value,
    uncertainty=0.0,
    reference='User',
    notes='',
    unit=Unit(dimensionless),
    raise_error=False,
)
Docstring:
Defines PhysicalData with uncertainty, reference and notes.

Note
----
It inherits from astropy.units.quantity.Quantity().

Parameters
----------
name :`str`
    The name representing the corresponding physical parameter.

value : `int`, `float`, `str`, `~numpy.ndarray`, `astropy.quantity.Quantity`
    The numerical value of this quantity in the units given by unit.  If
    a `Quantity`  (or any other valid object with a ``unit`` attribute),
    creates a new `Quantity` object, converting to `unit` units as needed.
    If a string, it is converted to a number or `Quantity`, depending on
    whether a unit is present.

uncertainty : `int`, `float`, `str`, `~numpy.ndarray`, `astropy.quantity.Quantity`,␣
↪default=0
    The numerical value of this quantity in the units given by unit.  If
    a `Quantity` (or any other valid object with a ``unit`` attribute),
    creates a new `Quantity` object, converting to `unit` units as needed.
```

(continues on next page)

(continued from previous page)

```
    If a string, it is converted to a number or `Quantity`, depending on
    whether a unit is present.

reference : `str`, default='User'
    A string stating the reference for the parameter value.

notes : `str`, default=''
    Any other important information about the physical parameter.

unit : `str`, `~astropy.units.UnitBase` instance, default='dimensionless'
    An object that represents the unit associated with the input value. Must
    be an `~astropy.units.UnitBase` object or a string parsable by the
    :mod:`~astropy.units` package.

raise_error : `bool`, default=False
    If ``value=None`` or ``raise_error=True`` the function raises an error,
    else `value` is redefined to ``NaN``.
File:           ~/Documentos/códigos/SORA/sora/body/meta.py
Type:           type
Subclasses:
```

[17]:
```
ceres.diameter = PhysicalData(name='Diameter', value=972, uncertainty=6, reference=
↪"Gomes-Júnior et al. (2015), MNRAS 451(3), 2295-2302",
                              notes="Equatorial Radius", unit=u.km)
print(ceres.diameter)
```

```
Diameter:
    972 +/- 6 km
    Reference: Gomes-Júnior et al. (2015), MNRAS 451(3), 2295-2302, Equatorial Radius
```

Some parameters have direct relations when modified, for instance: diameter and radius; and GM and mass.

[18]:
```
print(ceres.GM)
print(ceres.mass)
```

```
Standard Gravitational Parameter:
    62.628 +/- 0.0009 km3 / s2
    Reference: Nature vol. 537, pp515-517 (22 September 2016),

Mass:
    9.3835e+20 +/- 1.3485e+16 kg
    Reference: Nature vol. 537, pp515-517 (22 September 2016),
```

[19]:
```
ceres.GM = 63
ceres.GM.uncertainty = 0.0010
```

[20]:
```
print(ceres.GM)
print(ceres.mass)
```

```
Standard Gravitational Parameter:
    63 +/- 0.001 km3 / s2
    Reference: User,

Mass:
    9.4392e+20 +/- 1.4983e+16 kg
    Reference: User,
```

9.1.4 3. Ephemeris

An important feature of the body is its ephemeris. It is the ephemeris that will tell us te position of the object for a given time. It is necessary the Body object to have an Ephemeris when reducing an Occultation. The full description of the Ephemeris options are given at *Section 6*.

To add a ephemeris to a Body object, just given them as input or edit the ephem attribute:

[21]:
```
# Giving the kernels directly using an EphemKernel object
chariklo = Body(name='Chariklo', ephem=['input/bsp/Chariklo.bsp', 'input/bsp/de438_small.
↪bsp'])
```

```
Obtaining data for Chariklo from SBDB
```

[22]:
```
chariklo = Body(name='Chariklo')
chariklo.ephem = ['input/bsp/Chariklo.bsp', 'input/bsp/de438_small.bsp']
```

```
Obtaining data for Chariklo from SBDB
```

[23]:
```
print(chariklo)
```

```
###############################################################################
                          10199 Chariklo (1997 CU26)
###############################################################################
Object Orbital Class: Centaur
Spectral Type:
    SMASS: D  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Relatively featureless spectrum with very steep red slope.
Discovered 1997-Feb-15 by Spacewatch at Kitt Peak

Physical parameters:
Diameter:
    302 +/- 30 km
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),
Rotation:
    7.004 +/- 0 h
    Reference: LCDB (Rev. 2021-June); Warner et al., 2009, [Result based on less than␣
↪full coverage, so that the period may be wrong by 30 percent or so.]  REFERENCE LIST:
↪[Fornasier, S.; Lazzaro, D.; Alvarez-Candal, A.; Snodgrass, C.; et al. (2014) Astron.␣
↪Astrophys. 568, L11.], [Leiva, R.; Sicardy, B.; Camargo, J.I.B.; Desmars, J.; et al.␣
↪(2017) Astron. J. 154, A159.]
Absolute Magnitude:
    6.54 +/- 0 mag
    Reference: MPO691682,
```

(continues on next page)

(continued from previous page)

```
Albedo:
    0.045 +/- 0.01
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),

Ellipsoid: 151.0 x 151.0 x 151.0

---------- Ephemeris -----------

EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

[24]:
```
# Using the ephemeris from Horizons with an EphemHorizons object
ceres = Body(name='Ceres', ephem='horizons')
```

```
Obtaining data for Ceres from SBDB
```

The ephemeris parameters can also be an `EphemKernel`, `EphemHorizons` or `EphemPlanete` objects given directly. Only `EphemPlanete` requires the ephemeris object being defined before associated to a `Body` object.

[25]:
```
from sora import EphemKernel
ephem = EphemKernel(kernels=['input/bsp/Chariklo.bsp', 'input/bsp/de438_small.bsp'],␣
↪spkid='2010199')
chariklo.ephem = ephem
```

For a full description of the Ephemeris objects, see *Section 6*

9.1.5 4. Pole Position Angle and Aperture Angle

For a given time, and given pole coordinate, we are able to calculate the Pole Position Angle relative to the North pole in the ICRS and the Aperture Angle of the object pole coordinate. For this, it is necessary for the `Body` object to have pole coordinates and ephemeris.

[26]:
```
ceres.get_pole_position_angle?
```

```
Signature: ceres.get_pole_position_angle(time, observer='geocenter')
Docstring:
Returns the pole position angle and the aperture angle relative to
the geocenter.

Parameters
----------
time : `str`, `astropy.time.Time`
    Time from which to calculate the position.
    It can be a string in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time␣
↪object.

observer : `str`, `sora.Observer`, `sora.Spacecraft`
    IAU code of the observer (must be present in given list of kernels),
    a SORA observer object or a string: ['geocenter', 'barycenter']
```

(continues on next page)

(continued from previous page)

```
Returns
-------
position_angle, aperture_angle : `float` array
    Position angle and aperture angle of the object's pole, in degrees.
File:      ~/Documentos/códigos/SORA/sora/body/core.py
Type:      method
```

[27]:
```
ceres.pole.to_string('hmsdms')
```

[27]:
```
'19h25m40.32s +66d45m50.4s'
```

[28]:
```
pos_ang, ap_ang = ceres.get_pole_position_angle(time='2017-06-22 00:00:00')
print(pos_ang)
print(ap_ang)
```

```
348.5707051283039 deg
-2.5635901219177977 deg
```

9.1.6 5. Apparent Magnitude

The Body objectcs calculates the Apparent Magnitude of an object using the H and G given by the user. The equation is:

$\phi_0 = \exp(-3.33(\tan(0.5phase)^{0.63}))$

$\phi_1 = \exp(-1.87(\tan(0.5phase)^{1.22}))$

$apmag = H - 2.5\log 10((1-G)\phi_0 + G\phi_1) + 5\log 10(sundist * dist)$

It is Body object has the Absolute Magnitude (H), Phase Slope (G) and an ephemeris, the apparent magnitude will be calculated. Else, a query to the JPL Horizons web service will be made

[29]:
```
print(ceres.H, ceres.G)
```

```
Absolute Magnitude:
    3.33 +/- 0 mag
    Reference: MPO691494, IRAS observations used: 15
 Phase Slope:
    0.12 +/- 0
    Reference: PDS3 (MPC 17257), Fit
```

[30]:
```
ceres.apparent_magnitude(time='2017-06-22 00:00:00')
```

[30]:
```
9.011349951700899
```

[31]:
```
print(chariklo.H, chariklo.G)
```

```
Absolute Magnitude:
    6.54 +/- 0 mag
    Reference: MPO691682,
```

[32]:
```
chariklo.apparent_magnitude(time='2017-06-22 00:00:00')
```

```
/home/altair/Documentos/códigos/SORA/sora/body/core.py:332: UserWarning: H and/or G is␣
↪not defined for 10199 Chariklo. Searching into JPL Horizons service
  warnings.warn('H and/or G is not defined for {}. Searching into JPL Horizons service'.
↪format(self.shortname))
```

[32]:
```
18.489
```

When the apparent magnitude is downloaded from the Horizons service, usually the H and G parameters can be found. In this cases, the `Body` object stores these values for further use.

[33]:
```
print(chariklo.H, chariklo.G)
```

```
Absolute Magnitude:
    6.54 +/- 0 mag
    Reference: JPL Horizons,
 Phase Slope:
    0.15 +/- 0
    Reference: JPL Horizons,
```

[34]:
```
chariklo.apparent_magnitude(time='2017-06-22 00:00:00')
```

[34]:
```
18.489144139264898
```

9.1.7 6. The Ephemeris Classes

As stated above, there are 3 different types of `Ephemeris` Classes.

- `EphemPlanete` - Reads a file with a list of Geocentric coordinates and interpolates.
- `EphemHorizons` - Queries the JPL Horizons website and download ephemeris informations.
- `EphemKernel` - Reads the BSP files to calculate ephemeris.

Usually, the user will only need one of those. As shown in *Section 3*, the user can tell the body object the kernels for a `EphemKernel` object or the string 'horizons' for a `EphemHorizons` object. In this section we show the Ephemeris behave and if the user wants to use it directly without a `Body` object.

[35]:
```
## import the Ephemeris Classes
from sora.ephem import EphemPlanete, EphemHorizons, EphemKernel

## To facilitate, sora allows to import Ephem directly from the sora package.
from sora import EphemPlanete, EphemHorizons, EphemKernel
```

6.1. Instantiating an EphemPlanete Object

In this case, a file must be given containing Julian Date, RA (in deg), DEC (in deg) and Distance (in AU), in this order, separated by an space. Please, look at the "ephem_phoebe_ph15_jd.txt" file for an example. Only the name of the object and the file with the ephemeris is required.

[36]:
```
EphemPlanete?
```

```
Init signature: EphemPlanete(ephem, name=None, spkid=None, **kwargs)
Docstring:
Class used to simulate former Fortran programs `ephem_planete` and
`fit_d2_ksi_eta`.

Attributes
----------
ephem : `file`, required
    Input file with JD (UTC), geocentric RA (deg), DEC (deg), and
    distance (AU).

name : `str`, optional, default=None
    Name of the object to search in the JPL database.

radius : `int`, `float`, optional, default: online database
    Object radius, in km.

error_ra : `int`, `float`, optional, default: online database
    Ephemeris RA*cosDEC error, in arcsec.

error_dec : `int`, `float`, optional, default: online database
    Ephemeris DEC error, in arcsec.

mass : `int`, `float`, optional. default=0
    Object mass, in kg.

H : `int`, `float`, optional, default=NaN
    Object absolute magnitude.

G : `int`, `float`, optional, default=NaN
    Object phase slope.
File:           ~/Documentos/códigos/SORA/sora/ephem/core.py
Type:           type
Subclasses:
```

[37]:
```
eph_pla = EphemPlanete(name='Phoebe', ephem='input/ascii/ephem_phoebe_ph15_jd.txt')
```

[38]:
```
print(eph_pla)
```

```
----------- Ephemeris -----------

EphemPlanete: Valid from 2019-06-07 02:54:00.000 until 2019-06-07 04:54:00.000 (SPKID=)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

6.2. Instantiating an EphemHorizons Object

In this case, only the name of the object is required as this will be used to queries the JPL database. Another important keyword is the "id_type", which will be used to constrain the search of the object. For instance, if the name is "Europa" and the id_type is "smallbody" it looks the asteroid (52) Europa, while if id_type is "majorbody" it looks for the Jovian satellite J2 Europa. However, in some cases, there are still conflict name issue which solutions are being verified. A common solver is to use the IAU ID number in the name variable

[39]:
```
EphemHorizons?
```

```
Init signature: EphemHorizons(name, id_type='smallbody', spkid=None, **kwargs)
Docstring:
Obtains the ephemeris from Horizons/JPL service.

Note
----
Web tool URL: https://ssd.jpl.nasa.gov/horizons.cgi


Attributes
----------
name : `str`, required
    Name of the object to search in the JPL database.

id_type: `str`, default='smallbody'
    Type of object options: ``smallbody``, ``majorbody`` (planets but
    also anything that is not a small body), ``designation``, ``name``,
    ``asteroid_name``, ``comet_name``, ``id`` (Horizons id number), or
    ``smallbody`` (find the closest match under any id_type).

radius : `int`, `float`, default: online database
    Object radius, in km.

error_ra : `int`, `float`, default: online database
    Ephemeris RA*cosDEC error, in arcsec.

error_dec : `int`, `float`, default: online database
    Ephemeris DEC error, in arcsec.

mass : `int`, `float`, default=0
    Object mass, in kg.

H : `int`, `float`, default=NaN
    Object absolute magnitude.

G : `int`, `float`, default=NaN
    Object phase slope.
File:           ~/Documentos/códigos/SORA/sora/ephem/core.py
Type:           type
Subclasses:     EphemJPL
```

[40]:
```
eph_hor = EphemHorizons(name='Chariklo')
```

[41]:
```
print(eph_hor)
```

```
----------- Ephemeris -----------

EphemHorizons: Ephemeris are downloaded from Horizons website (SPKID=)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

6.3. Instantiating an EphemKernel Object

This is the most effective `Ephemeris` Class. Since it reads the BSP files directly, it is faster. What is needed here is the spkid of the object, and the list of kernels. Even if only one kernel is passed, it must be a list.

[42]:
```
EphemKernel?
```

```
Init signature: EphemKernel(kernels, spkid, name=None, **kwargs)
Docstring:
Gets the ephemeris from BSP kernels.

Parameters
----------
name : `str`,  optional, default=None
    Name of the object to search in the JPL database.

spkid : `str`, required
    `spkid` of the targeting object. Former 'code' (v0.1).

kernels : `list`, required
    List of paths for kernels files.

radius : `int`, `float`, optional, default: online database
    Object radius, in km.

error_ra : `int`, `float`, optional, default: online database
    Ephemeris RA*cosDEC error, in arcsec .

error_dec : `int`, `float`, optional, default: online database
    Ephemeris DEC error, in arcsec.

mass : `int`, `float`, optional, default=0
    Object Mass, in kg.

H : `int`, `float`, optional, default=NaN
    Object Absolute Magnitude.

G : `int`, `float`, optional, default=NaN
    Object Phase slope.
File:           ~/Documentos/códigos/SORA/sora/ephem/core.py
Type:           type
Subclasses:
```

[43]:
```
eph_ker = EphemKernel(name='Chariklo', spkid='2010199', kernels=['input/bsp/Chariklo.bsp
↪', 'input/bsp/de438_small.bsp'])
```

[44]:
```
print(eph_ker)
```

```
----------- Ephemeris -----------

EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

6.4 Other parameters

All the 3 `Ephemeris` Classes allow for the user to pass the ephemeris error in RA*cosDEC and DEC.

[45]:
```
eph_ker = EphemKernel(name='Chariklo', spkid='2010199', kernels=['input/bsp/Chariklo.bsp
↪', 'input/bsp/de438_small.bsp'], error_ra=0.02, error_dec=0.02)
```

[46]:
```
print(eph_ker)
```

```
----------- Ephemeris -----------

EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.020 arcsec; DEC: 0.020 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

6.5. Obtaining positions

The `Ephemeris` allows the user to get the astrometric position of the object for a certain epoch. The returned variable is an Astropy SkyCoord object, so their functions can be used.

[47]:
```
chariklo.ephem.get_position?
```

```
Signature: chariklo.ephem.get_position(time, observer='geocenter')
Docstring:
Returns the object geocentric position.

Parameters
----------
time : `str`, `astropy.time.Time`
    Reference time to calculate the object position. It can be a string
    in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

observer : `str`, `sora.Observer`, `sora.Spacecraft`
    IAU code of the observer (must be present in given list of kernels),
    a SORA observer object or a string: ['geocenter', 'barycenter']

Returns
-------
```

(continues on next page)

(continued from previous page)

```
coord : `astropy.coordinates.SkyCoord`
    Astropy SkyCoord object with the object coordinates at the given time.
File:      ~/Documentos/códigos/SORA/sora/ephem/core.py
Type:      method
```

[48]:
```
pos = chariklo.ephem.get_position(time='2017-06-22 00:00:00', observer='geocenter')
print(pos)
```

```
<SkyCoord (ICRS): (ra, dec, distance) in (deg, deg, km)
    (283.86685123, -31.52237927, 2.19331444e+09)>
```

[49]:
```
print('RA =', pos.ra)
print('DEC =', pos.dec)
print('Distance =', pos.distance.to('AU'))
print('Coordinate =', pos.to_string('hmsdms', precision=5, sep=' '))
```

```
RA = 283d52m00.66441732s
DEC = -31d31m20.56535577s
Distance = 14.661401433721714 AU
Coordinate = 18 55 28.04429 -31 31 20.56536
```

[50]:
```
eph_hor.get_position(time='2017-06-22 00:00:00', observer='barycenter')
```

[50]:
```
<SkyCoord (ICRS): (ra, dec, distance) in (deg, deg, AU)
    (282.96780287, -31.0449271, 15.6440208)>
```

The astrometric position can also be obtained for a specific observer.

This is restricted for the EphemPlanete Class since the input coordinates already have an observer associated.

[51]:
```
from sora import Observer
opd = Observer(name='Pico dos Dias Observatory', code='874')

chariklo.ephem.get_position(time='2017-06-22 00:00:00', observer=opd)
```

[51]:
```
<SkyCoord (ICRS): (ra, dec, distance) in (deg, deg, km)
    (283.86701022, -31.52236659, 2.19331059e+09)>
```

6.6. Ksi and Eta projection

`Ephemeris` objects calculates the orthographic projection (ksi and eta) of a site in the direction of a star given the following function. Ksi is in the East direction and Eta is in the North direction.

The coordinates of the star given must be in the Geocentric Celestial Reference System (GCRS).

This is calculated automaticaly in `Occultation`.

[52]:
```
chariklo.ephem.get_ksi_eta?
```

```
Signature: chariklo.ephem.get_ksi_eta(time, star)
Docstring:
Returns the object's projected position relative to a star.
```

(continues on next page)

(continued from previous page)

```
Returns the projected position (orthographic projection) of the object
in the tangent sky plane relative to a star.

Parameters
----------
time : `str`, `astropy.time.Time`
    Reference time to calculate the object position. It can be a string
    in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

star : `str`, `astropy.coordinates.SkyCoord`
    Coordinate of the star in the same reference frame as the ephemeris.

Returns
-------
ksi, eta : `float`
    Projected position (orthographic projection) of the object in the
    tangent sky plane relative to a star.
    ``ksi`` is in the North-South direction (North positive).
    ``eta`` is in the East-West direction (East positive).
File:      ~/Documentos/códigos/SORA/sora/ephem/meta.py
Type:      method
```

[53]:
```
chariklo.ephem.get_ksi_eta(time='2017-06-22 21:18:00', star='18 55 15.65250 -31 31 21.
↪67051')
```

[53]:
```
(1059.3876121640205, 522.2713452577591)
```

[54]:
```
eph_hor.get_ksi_eta(time='2017-06-22 21:18:00', star='18 55 15.65250 -31 31 21.67051')
```

[54]:
```
(1014.75804445542, 645.688436361363)
```

In the case of `EphemPlanete` , the time given must be in the range of the input ephemeris.

[55]:
```
print(eph_pla)
```

```
----------- Ephemeris -----------

EphemPlanete: Valid from 2019-06-07 02:54:00.000 until 2019-06-07 04:54:00.000 (SPKID=)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

[56]:
```
eph_pla.get_ksi_eta(time='2019-06-07 03:54:22.60', star='19 21 18.63201 -21 44 25.3924')
```

```
Fitting ephemeris position relative to star coordinate 19h21m18.63201s -21d44m25.3924s
ksi = aksi*t² + bksi*t + cksi
eta = aeta*t² + beta*t + ceta
t=(jd-2458641.62083333)/(2458641.70416667-2458641.62083333)
        aksi=-74.33578570357841
        bksi=-120036.66964478235
        cksi=56728.18870944259
        aeta=-11.579369275877033
```

(continues on next page)

(continued from previous page)

```
        beta=-17144.75884811865
        ceta=6972.3299907265955
Residual RMS: ksi=0.004 km, eta=0.001 km
```

[56]:
```
(-3685.7460633119117, -1656.79625359093)
```

For `EphemPlanete` , the user can also make the fit before running get_ksi_eta. If the fit is done, get_ksi_eta will only need the time param.

[57]:
```
eph_pla.fit_d2_ksi_eta?
```

```
Signature: eph_pla.fit_d2_ksi_eta(star, verbose=True)
Docstring:
Fits the projected position (orthographic projection) of the object in
the tangent sky plane relative to a star.

Parameters
----------
star : `str`, `astropy.coordinates.SkyCoord`
    The coordinate of the star in the same reference frame as the ephemeris.

verbose : `bool`, optional, default=True
    Enable log printing.
File:      ~/Documentos/códigos/SORA/sora/ephem/core.py
Type:      method
```

[58]:
```
eph_pla.fit_d2_ksi_eta(star='19 21 18.63201 -21 44 25.3923')
```

```
Fitting ephemeris position relative to star coordinate 19h21m18.63201s -21d44m25.3923s
ksi = aksi*t² + bksi*t + cksi
eta = aeta*t² + beta*t + ceta
t=(jd-2458641.62083333)/(2458641.70416667-2458641.62083333)
        aksi=-74.33578570357841
        bksi=-120036.66964478235
        cksi=56728.18870944259
        aeta=-11.579369458928182
        beta=-17144.75878814249
        ceta=6971.659735843645
Residual RMS: ksi=0.004 km, eta=0.001 km
```

[59]:
```
eph_pla.get_ksi_eta(time='2019-06-07 03:54:22.60')
```

[59]:
```
(-3685.7460633119117, -1657.4664783438802)
```

6.7. Adding Offsets

If an ephemeris offset was obtained from astrometry, it can be added to the `Ephemeris` objects using add_offset. This values are given in $\Delta\alpha \cos\delta$ and $\Delta\delta$ in mas.

[60]:
```
eph_ker.add_offset?
```

```
Signature: eph_ker.add_offset(da_cosdec, ddec)
Docstring:
Adds an offset to the Ephemeris.

Parameters
----------
da_cosdec : `int`, `float`
    Delta_alpha_cos_delta, in mas.

ddec : `int`, `float`
    Delta_delta, in mas.
File:      ~/Documentos/códigos/SORA/sora/ephem/meta.py
Type:      method
```

[61]:
```
print(chariklo.ephem.get_position(time='2017-06-22 00:00:00').to_string('hmsdms'))
eph = chariklo.ephem
```

```
18h55m28.04429449s -31d31m20.56535577s
```

[62]:
```
eph.add_offset(40, 50)
eph_hor.add_offset(40, 50)
eph_pla.add_offset(40, 50)
```

[63]:
```
eph.get_position(time='2017-06-22 00:00:00').to_string('hmsdms')
```

[63]:
```
'18h55m28.04742278s -31d31m20.51535577s'
```

9.1.8 7. The frame attribute

For some bodies, we know its rotational parameters and polar coordinates. In this cases, we may need to obtain its rotational phase at a specific epoch.

When we print the body object, we may see a parameter called `PlanetocentricFrame`. This is the name of class responsible for the computation of the body's orientation. The calculations are based on the most recent Report of the IAU Working Group on Cartographic Coordinates and Rotational Elements. For the bodies which parameters are present in the report, like Ceres, the frame is automatically defined in the Body object. For other objects, it may be necessary for the user to define it.

The equations will not be reproduced here. Please refer to the report for further details.

[64]:
```
print(ceres.frame)
```

```
PlanetocentricFrame:
    Epoch: J2000.000
    alpha_pole = 291.418 +0.000000*T +0.000000
    delta_pole = 66.764 +0.000000*T +0.000000
```

(continues on next page)

(continued from previous page)

```
    W = 170.65 +952.153200*d +0.000000
    Reference: Archinal, B. A., Report of the IAU Working Group on Cartographic␣
↪Coordinates and Rotational Elements: 2015, Celestial Mechanics and Dynamical Astronomy,
↪ (2018) 130:22
```

The frame can be obtained for a different epoch. In this case, the body will be precessed and rotated to the referred epoch (if parameters are available)

[65]:
```
print(ceres.frame.frame_at(epoch='2022-09-20 00:00:00'))
```

```
PlanetocentricFrame:
    Epoch: 2022-09-20 00:00:00.000
    alpha_pole = 291.418 +0.000000*T +0.000000
    delta_pole = 66.764 +0.000000*T +0.000000
    W = 102.58942676521838 +952.153200*d +0.000000
    Reference: Archinal, B. A., Report of the IAU Working Group on Cartographic␣
↪Coordinates and Rotational Elements: 2015, Celestial Mechanics and Dynamical Astronomy,
↪ (2018) 130:22
```

To obtain the the orientation as viewed by an observer, we must use the `get_orientation()` function on the body object. It will determine the ephemeris of the body, calculate the light time and compute the orientation

Important Note: The subobserver latitude and longitude are PLANETOCENTRIC, and not PLANETODETIC.

[66]:
```
ceres.get_orientation(time='2022-09-20 00:00:00', observer='geocenter')
```

[66]:
```
{'sub_observer': '137.824 -0.567677',
 'sub_solar': '137.824 -0.567158',
 'pole_position_angle': <Quantity 14.30611673 deg>,
 'pole_aperture_angle': <Quantity -0.56767665 deg>}
```

9.1.9 8. The shape attribute

Since the version 0.3 of SORA the shape attribute was included. With it, the user can provide a 3D shape object or define an ellipsoid. Thus, it is possible to plot the shape of the object as viewed by an observer.

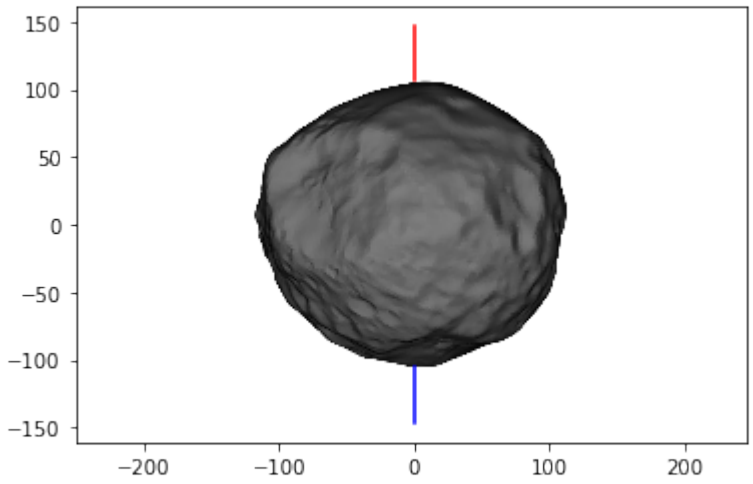
To replicate the process shown below, it necessary to download the 3D shape model of Phoebe here. The file must be the in "OBJ" format. Either the 50k or 200k versions will work, however the 200k one will demand more processing. The surface texture can also be downloaded and included in the process, but it is not relevant for the fitting process, only for plottings.

[67]:
```
phoebe = Body(name="Phoebe", shape="Phoebe_Gaskell_50k_poly.obj")
```

[68]:
```
print(phoebe.shape)
```

```
Shape3D: Phoebe_Gaskell_50k_poly.obj
```

[69]:
```
# A simple plot of the shape will show a view from lat=0, lon=0
phoebe.shape.plot()
```



We can provide the coordinates we want to view the object. Notice the red and blue lines. They show the north and south poles, respectively.

[70]:
```
phoebe.shape.plot?
```

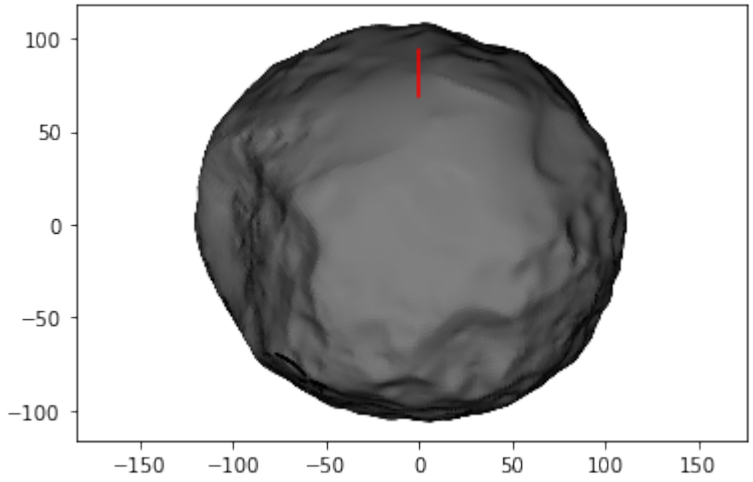
```
Signature:
phoebe.shape.plot(
    sub_observer='00 00 00 +00 00 00',
    sub_solar=None,
    pole_position_angle=0,
    center_f=0,
    center_g=0,
    scale=1,
    ax=None,
    plot_pole=True,
    **kwargs,
)
Docstring:
Parameters
----------
sub_observer : `astropy.coordinates.SkyCoord`, `str`
    Planetocentric coordinates of the center of the object in the direction of the␣
↪observer.
    It can be an astropy SkyCoord object or a string with the bodycentric longitude
    latitude in degrees. Ex: "30.0 -20.0", or "30 00 00 -20 00 00".

sub_solar : `astropy.coordinates.SkyCoord`, `str`
    Planetocentric coordinates of the center of the object in the direction of the Sun.
    It can be an astropy SkyCoord object or a string with the bodycentric longitude
    latitude in degrees. Ex: "30.0 -20.0", or "30 00 00 -20 00 00".

pole_position_angle : `float`, `int`
    Body's North Pole position angle with respect to direction of the ICRS
    North Pole, i.e. N-E-S-W.

center_f : `int`, `float`
    Offset of the center of the body in the East direction, in km
```

(continues on next page)

(continued from previous page)

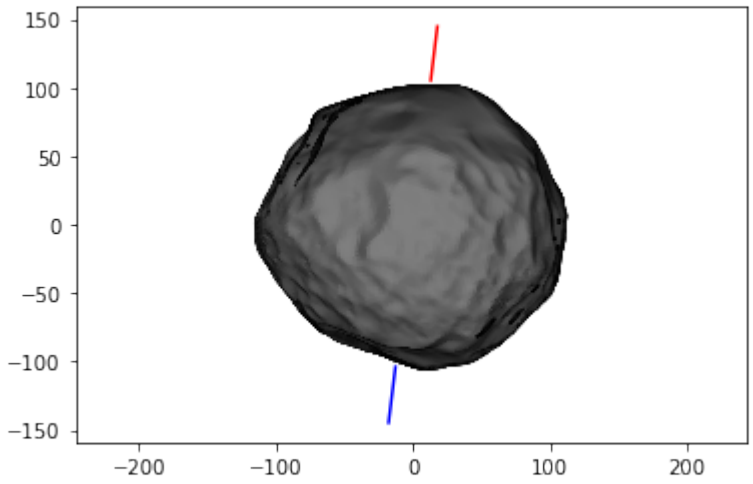
```
center_g  : `int`, `float`
    Offset of the center of the body in the North direction, in km

scale : `float`
    Multiply the shape vertices by a value. Default=1

ax : `matplotlib.pyplot.Axes`
    The axes where to make the plot. If None, it will use the default axes.

plot_pole : `bool`
    If True, the direction of the pole is plotted.

**kwargs
    Any other keyword argument is discarded
File:      ~/Documentos/códigos/SORA/sora/body/shape/core.py
Type:      method
```

[71]:
```
phoebe.shape.plot(sub_observer="30.0 +50.0")
```



We can also plot the shape as viewed by an observer at a specific time. For this, we must plot the body directly, where the frame attribute will be called to compute the correct orientation.

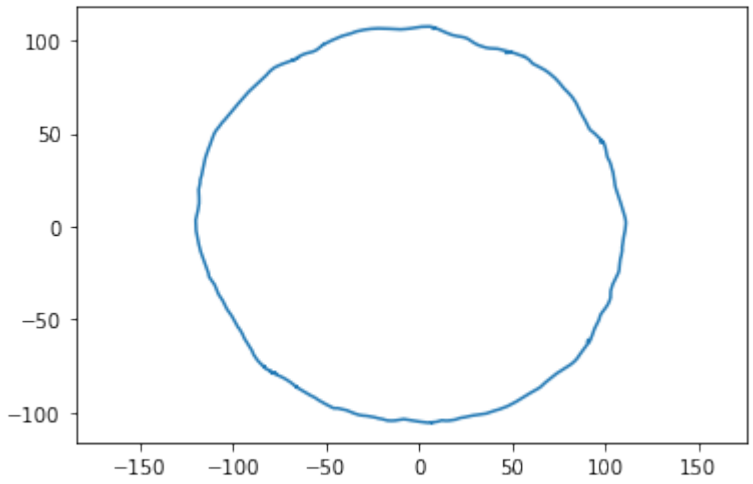
[72]:
```
phoebe.plot(time='2022-09-20', observer='geocenter')
```



To use the 3D shape in the occultation process, we need to be able to get the projected limb of the 3D shape. For that we can use the `get_limb` function in the shape attribute

[73]:
```
limb = phoebe.shape.get_limb(sub_observer="30.0 +50.0")
```

[74]:
```
limb.plot()
```



To see the use of the shape in an occultation, there is an example in our online documentation showing all the steps.

This Jupyter-Notebook was designed as a tutorial for how to work with the `Body` and `Ephemeris` Classes. More information about the other classes, please refer to their specif Jupyter-Notebook. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The End

9.2 Star Class

The `Star` Class within SORA was created to deal with the star information. The documentation here contains the details about every step.

This Jupyter-Notebook was designed as a tutorial for how to work with the `Star` Class. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The `Star` Docstring was designed to help the users. Also, each function has its Docstring containing its main purpose and the needed parameters (physical description and formats). Please, do not hesitate to use it.

9.2.1 0. Index

[1]:
```
## import the Star Class
from sora.star import Star

## To facilitate, sora allows to import Star directly from the root
from sora import Star
```

```
SORA version: 0.3
```

9.2.2 1. Instantiating Star Object

The `Star` Class can be instantiated in different ways. Using the Gaia-DR2 or Gaia-EDR3 Source ID, then all the informations of the stars (RA, DEC, parallax, proper motions, G magnitude and star radius) will be downloaded from VizieR. Using a coordinate, then the Gaia-EDR3 (or Gaia-DR2) star will be searched in VizieR within a radius of 2 arcsec. If more than 1 star is found, the user will be asked to choose the correct one.

`Star` can also download the B, V, R, J, H and K magnitudes from the NOMAD catalogue. This is useful for calculating the apparent diameter of a star using van Belle's or Kevella's methods. If more than 1 star is found within 2 arcsec from the given coordinate, the user will be asked to choose the correct one.

[2]:
```
Star?
```

```
Init signature: Star(catalogue='gaiadr3', **kwargs)
Docstring:
Defines a star.

Parameters
----------
catalogue : `str`, `VizierCatalogue`
    The catalogue to download data. It can be ``'gaiadr2'``, ``'gaiaedr3'``,
    ``'gaiadr3'``, or a VizierCatalogue object.. default='gaiadr3'
```

(continues on next page)

(continued from previous page)

```
code : `str`
    Gaia Source code for searching in VizieR.

coord : `str`, `astropy.coordinates.SkyCoord`
    If code is not given, coord nust have the coordinates RA and DEC of the
    star to search in VizieR: ``'hh mm ss.ss +dd mm ss.ss'``.

ra : `int`, `float`
    Right Ascension, in deg.

dec : `int`, `float`
    Declination, in deg.

parallax : `int`, `float`. default=0
    Parallax, in mas.

pmra : `int`, `float`, default=0
    Proper Motion in RA*, in mas/year.

pmdec : `int`, `float`, default=0
    Proper Motion in DEC, in mas/year.

rad_vel : `int`, `float`, default=0
    Radial Velocity, in km/s.

epoch : `str`, `astropy.time.Time`, default='J2000'
    Epoch of the coordinates.

nomad : `bool`
    If True, it tries to download the magnitudes from NOMAD catalogue.

bjones : `bool`, default=True
    If True, it uses de star distance from Bailer-Jones et al. (2018).

cgaudin : `bool`, default=True
    If True, it uses de proper motion correction from Cantat-Gaudin & Brandt (2021).
    this option is only available for Gaia-EDR3.

verbose : `bool`, default=True
    If True, it prints the downloaded information

local : `bool`, default=False
    If True, it uses the given coordinate in 'coord' as final coordinate.

Note
----
The user can give either 'coord' or 'ra' and 'dec', but not both.

To download the coordinates from Gaia, "local" must be set as False
and the ("code") or ("coord") or ("ra" and "dec") must be given.

All values downloaded from Gaia will replace the ones given by the user.
```

(continues on next page)

(continued from previous page)

```
File:           ~/Documentos/códigos/SORA/sora/star/core.py
Type:           type
Subclasses:
```

Searching by Gaia Source ID

[3]:
```
star = Star(code='6306109685386306688')
```

```
1 GaiaDR3 star found band={'G': 15.341292}
star coordinate at J2016.0: RA=15h04m17.67141s +/- 0.0307 mas, DEC=-16d19m38.9718s +/- 0.
↪0241 mas

Downloading star parameters from I/297/out
```

[4]:
```
print(star)
```

```
GaiaDR3 star Source ID: 6306109685386306688
ICRS star coordinate at J2016.0:
RA=15h04m17.67141s +/- 0.0307 mas, DEC=-16d19m38.9718s +/- 0.0241 mas
pmRA=6.286 +/- 0.040 mas/yr, pmDEC=-9.632 +/- 0.033 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.8595 +/- 0.0332 mas, Rad. Vel.=0.00 +/- 0.00 km/s

Magnitudes: G: 15.341, B: 15.460, V: 14.680, R: 15.240, J: 14.121, H: 13.773,
            K: 13.643

Apparent diameter from Kervella et. al (2004):
    V: 0.0074 mas, B: 0.0077 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0090 mas, V: 0.0092 mas
    ms: B: 0.0086 mas, V: 0.0069 mas
    vs: B: 0.0135 mas, V: 0.0120 mas
```

Searching by coordinate

[5]:
```
star2 = Star(coord='18 22 00.25777 -22 28 10.7057')
```

```
2 stars were found within 2 arcsec from given coordinate.
The list below is sorted by distance. Please select the correct star
num dist(")  Gmag         RA___ICRS___DEC
--- ------- ------ ------------------------------
  1   0.022 15.589 18h22m00.2584s -22d28m10.6860s
  2   2.009 19.981 18h22m00.3855s -22d28m09.7559s
  0: Cancel
```

```
Choose the corresponding number of the correct star:  1
```

```
1 GaiaDR3 star found band={'G': 15.589052}
star coordinate at J2016.0: RA=18h22m00.25843s +/- 0.0357 mas, DEC=-22d28m10.6860s +/- 0.
↪0313 mas

Downloading star parameters from I/297/out
2 stars were found within 2 arcsec from given coordinate.
The list below is sorted by distance. Please select the correct star
```

(continues on next page)

(continued from previous page)

```
num dist(")  Bmag   Vmag   Rmag   Jmag   Hmag   Kmag         RA___ICRS___DEC
--- ------- ------ ------ ------ ------ ------ ------ ------------------------------
  1   0.718 17.600    nan    nan    nan    nan    nan 18h22m00.2627s -22d28m09.9700s
  2   1.774 18.330    nan 14.500 12.419 11.471 11.128 18h22m00.1313s -22d28m10.8901s
  0: Cancel
```

```
Choose the corresponding number of the correct star:  1
```

```
Magnitudes in ['V', 'R', 'J', 'H', 'K'] were not located in NOMAD
```

[6]:
```
print(star2)
```

```
GaiaDR3 star Source ID: 4089802618196107776
ICRS star coordinate at J2016.0:
RA=18h22m00.25843s +/- 0.0357 mas, DEC=-22d28m10.6860s +/- 0.0313 mas
pmRA=-3.564 +/- 0.045 mas/yr, pmDEC=-8.035 +/- 0.033 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.1181 +/- 0.0377 mas, Rad. Vel.=-79.08 +/- 7.97 km/s

Magnitudes: G: 15.589, B: 17.600
```

Note that there are warnings if it does not find some information (in this case, the magnitudes from NOMAD)

By default, the Star object will search by a Gaia-DR3 star. However, the user can also set that Gaia-DR2 catalogue should be used instead.

[7]:
```
star_3 = Star(code='6306109685386306688', catalogue='gaiadr2')
```

```
1 GaiaDR2 star found band={'G': 15.3528}
star coordinate at J2015.5: RA=15h04m17.67119s +/- 0.0432 mas, DEC=-16d19m38.9668s +/- 0.
↪0309 mas

Downloading star parameters from I/297/out
```

[8]:
```
s = Star(coord='15 04 17.6 -16 19 38.9')
```

```
1 GaiaDR3 star found band={'G': 15.341292}
star coordinate at J2016.0: RA=15h04m17.67141s +/- 0.0307 mas, DEC=-16d19m38.9718s +/- 0.
↪0241 mas

Downloading star parameters from I/297/out
```

[9]:
```
print(s)
```

```
GaiaDR3 star Source ID: 6306109685386306688
ICRS star coordinate at J2016.0:
RA=15h04m17.67141s +/- 0.0307 mas, DEC=-16d19m38.9718s +/- 0.0241 mas
pmRA=6.286 +/- 0.040 mas/yr, pmDEC=-9.632 +/- 0.033 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.8595 +/- 0.0332 mas, Rad. Vel.=0.00 +/- 0.00 km/s

Magnitudes: G: 15.341, B: 15.460, V: 14.680, R: 15.240, J: 14.121, H: 13.773,
            K: 13.643
```

(continues on next page)

(continued from previous page)

```
Apparent diameter from Kervella et. al (2004):
    V: 0.0074 mas, B: 0.0077 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0090 mas, V: 0.0092 mas
    ms: B: 0.0086 mas, V: 0.0069 mas
    vs: B: 0.0135 mas, V: 0.0120 mas
```

9.2.3 2. Changing or including magnitudes

If one necessary magnituge in a specific band is not found in NOMAD or the user wants to add a different value to be saved in the Star history, it can be done with set_magnitude

[10]:
```
star.set_magnitude?
```

```
Signature: star.set_magnitude(**kwargs)
Docstring:
Sets the magnitudes of a star.

Parameters
----------
band=value : `str`
    The star magnitude for given band. The band name can be any string
    the user wants.

Examples
--------
To set the stars magnitude in the band G:

>>> set_magnitude(G=10)

To set the star's magnitude in the band K:

>>> set_magnitude(K=15)

To set the star's magnitude in a customized band:

>>> set_magnitude(newband=6)
File:      ~/Documentos/códigos/SORA/sora/star/core.py
Type:      method
```

[11]:
```
# Changing an already existing band will show a warning, but the change is done.
star.set_magnitude(V=14)
```

```
/home/altair/Documentos/códigos/SORA/sora/star/core.py:153: UserWarning: V mag already␣
↪defined. V=14.680000305175781 will be replaced by V=14.0
  warnings.warn('{0} mag already defined. {0}={1} will be replaced by {0}={2}'.format(
```

[12]:
```
star.mag
```

[12]:
```
{'G': 15.341292,
 'B': 15.460000038146973,
 'V': 14.0,
 'R': 15.239999771118164,
 'J': 14.121000289916992,
 'H': 13.77299976348877,
 'K': 13.642999649047852}
```

set_magnitude does not have any pre-set band name, so the user can give whatever the name of the band. It must a string though.

[13]:
```
star2.set_magnitude(x_ray=15, ultraviolet=10)
```

[14]:
```
star2.mag
```

[14]:
```
{'G': 15.589052, 'B': 17.600000381469727, 'x_ray': 15.0, 'ultraviolet': 10.0}
```

9.2.4 3. Propagated positions

If the proper motion and parallax is found in the Gaia catalogue, the user can see the ICRS coordinate of the star at any epoch. It can be barycentric (corrected from proper motion), geocentric (corrected from proper motion and parallax), or for any Observer object.

The returned variable is an Astropy `SkyCoord` object.

[15]:
```
# The coord attribute shows the values kept for the star in RA, DEC, distance, proper␣
↪motion and radial velocity
print(star2.coord)
```

```
<SkyCoord (ICRS): (ra, dec, distance) in (deg, deg, pc)
    (275.50107681, -22.46963499, 8467.40050804)>
```

[16]:
```
print(star2.coord.to_string('hmsdms'))
```

```
18h22m00.25843459s -22d28m10.68596368s
```

[17]:
```
star2.get_position?
```

```
Signature: star2.get_position(time, observer='geocenter')
Docstring:
Calculates the position of the star for given observer,
propagating the position using parallax and proper motion

Parameters
----------
time : `float`, `astropy.time.Time`
    Reference time to apply proper motion and calculate parallax. It can be a string
    in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

observer : `str`, `sora.observer.Observer`, `sora.observer.Spacecraft`
    Observer of the star t calculate position. It can be 'geocenter' for a geocentric
    coordinate, 'barycenter' for a barycenter coordinate, or a sora observer object.
```

(continues on next page)

(continued from previous page)

```
Returns
-------
coord : `astropy.coordinates.SkyCoord`
    Astropy SkyCoord object with the star coordinates at the given time.
File:       ~/Documentos/códigos/SORA/sora/star/core.py
Type:       method
```

[18]:
```
pos = star2.get_position(time='2020-05-15 00:00:00', observer='barycenter')
print(pos.to_string('hmsdms', precision=10))
```

```
18h22m00.2573114293s -22d28m10.7210625154s
```

[19]:
```
pos = star2.get_position(time='2020-05-15 00:00:00', observer='geocenter')
print(pos.to_string('hmsdms', precision=10))
```

```
18h22m00.2573170776s -22d28m10.7210581733s
```

[20]:
```
from sora import Observer
opd = Observer(name='Pico dos Dias Observatory', code='874')

pos = star2.get_position(time='2020-05-15 00:00:00', observer=opd)
print(pos.to_string('hmsdms', precision=10))
```

```
18h22m00.2573170779s -22d28m10.7210581716s
```

The position error projected at epoch can also be called out. This is done using the formalism proposed by Butkevich et al., 2014, using the parameters and covariance matrix from Gaia catalogue (EDR3 or DR2).

[21]:
```
star2.error_at(time='2020-05-15 00:00:00')
```

[21]:
```
(<Quantity 0.14513241 mas>, <Quantity 0.15681425 mas>)
```

9.2.5 4. Adding offsets

The Gaia coordinate is expected to be the most precise coordinate published so far. However, if the user has the need to add an offset to the coordinate, it can be done with add_offset. The input values must be in $\Delta\alpha \cos\delta$ and $\Delta\delta$, in mas. The application of the offset is done only in the geocentric function.

[22]:
```
star2.get_position(time='2020-05-15 00:00:00', observer='geocenter').to_string('hmsdms')
```

[22]:
```
'18h22m00.25731708s -22d28m10.72105817s'
```

[23]:
```
star2.add_offset(40,50)
```

[24]:
```
star2.get_position(time='2020-05-15 00:00:00', observer='geocenter').to_string('hmsdms')
```

[24]:
```
'18h22m00.26020282s -22d28m10.67105817s'
```

9.2.6 5. The Apparent Diameter of the star

SORA is able to calculate the apparent diameter of a star at a given distance using the star radius of the Gaia catalogue or the angular diameter calculated from the methods of Kervella (2004) or van Belle (1999). These functions can be used alone, or within the `Star` object.

[25]:
```
from sora.star import kervella, van_belle
```

The functions will return the diameter calculated for B and V magnitudes, in mas

[26]:
```
kervella(magB=10, magV=11, magK=10)
```

[26]:
```
{'V': <Quantity 0.03912911 mas>, 'B': <Quantity 0.03280198 mas>}
```

In the case of van_belle function, the result is given for Super Giant "sg", Main Sequence "ms" and Variable Star "vs"

[27]:
```
van_belle(magB=10, magV=11, magK=10)
```

[27]:
```
{'sg': {'B': <Quantity 0.04446313 mas>, 'V': <Quantity 0.04920395 mas>},
 'ms': {'B': <Quantity 0.03162278 mas>, 'V': <Quantity 0.03664376 mas>},
 'vs': {'B': <Quantity 0.0691831 mas>, 'V': <Quantity 0.06412096 mas>}}
```

If a Star object is defined and the B, V and K magnitudes are defined, it can be directly called as:

[28]:
```
star.kervella()
```

[28]:
```
{'V': <Quantity 0.00653663 mas>, 'B': <Quantity 0.00766495 mas>}
```

[29]:
```
star.van_belle()
```

[29]:
```
{'sg': {'B': <Quantity 0.00903109 mas>, 'V': <Quantity 0.00888405 mas>},
 'ms': {'B': <Quantity 0.00860855 mas>, 'V': <Quantity 0.00622656 mas>},
 'vs': {'B': <Quantity 0.01353278 mas>, 'V': <Quantity 0.01166342 mas>}}
```

For a `Star` object, to see the apparent diameter, in km, at a given distance, the apparent_diameter method can be used.

If no parameter is given, the source will be automatically chosen given the following sequence:

```
- A value given by the user
- The star radius obtained from Gaia.
- The apparent diameter calculated from Kervella at V band
- The apparent diameter calculated from van Belle at V band for a Super Giant star.
```

[30]:
```
star.apparent_diameter?
```

```
Signature:
star.apparent_diameter(
    distance,
    mode='auto',
    band='V',
    star_type='sg',
    verbose=True,
)
Docstring:
```

(continues on next page)

(continued from previous page)

```
Calculates the apparent diameter of the star at a given distance.

Parameters
----------
distance : `int`, `float`
    Object geocentric distance, in AU.

mode : `str`, default='auto'
    The mode to calculate the apparent diameter.

    - ``'user'``: calculates using user given diameter.

    - ``'gaia'``: calculates using diameter obtained from Gaia.

    - ``'kervella'``: calculates using Kervella equations.

    - ``'van_belle'``: calculates using van Belle equations.

    - ``'auto'``: tries all the above methods until it is able to calculate diameter.

    The order of try is the same as shown above (user, Gaia, Kervella, Van Belle).

band : `str`
    The band filter to calculate the diameter. If mode is `kervella`
    or `van_belle`, the filter must be given, ``'B'`` or ``'V'``.
    If mode `auto`, ``'V'`` is selected.

star_type :`str`
    Type of star to calculate the diameter. If mode is `van_belle`,
    the star type must be given. If mode is `auto`, ``star_type='sg'``.

    Accepted types:

    - ``'sg'`` for 'Super Giant'.

    - ``'ms'`` for 'Main Sequence'.

    - ``'vs'`` for 'Variable Star'.

verbose : `bool`
    If True, it prints the mode used by `auto`.
File:      ~/Documentos/códigos/SORA/sora/star/core.py
Type:      method
```

[31]:
```
star.apparent_diameter(distance=9) #Distance in AU
```

```
Apparent diameter using Kervella et al. (2004)
```

[31]: 0.04266742 km

To set an user diameter, in mas

[32]:
```
star.set_diameter(diameter=0.05)
```

[33]:
```
star.apparent_diameter(distance=9)
```

```
Calculating apparent diameter from user defined diameter
```

[33]: 0.32637192 km

To choose the source of calculation, just select in the mode parameter. If `mode='kervella'`, the user must give 'band' as 'B or 'V'. If `mode='van_belle'`, the user must give the band and the type of star in star_type as 'sg', 'ms', or 'vs'. If not given, `band='V'` and `star_type='sg'`.

[34]:
```
star.apparent_diameter(distance=9.5, mode='kervella')
```

```
Apparent diameter using Kervella et al. (2004)
```

[34]: 0.045037832 km

[35]:
```
star.apparent_diameter(distance=9.5, mode='van_belle', band='B', star_type='ms')
```

```
Apparent diameter using van Belle (1999)
```

[35]: 0.05931356 km

This Jupyter-Notebook was designed as a tutorial for how to work with the `Star` Class. More information about the other classes, please refer to their specif Jupyter-Notebook. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The End

9.3 *prediction()* function and the PredictionTable Class

The prediction module within SORA was created to predict stellar occultation, organize the prediction and plot the occultation maps. The documentation here contains the details about every step.

This Jupyter-Notebook was designed as a tutorial for how to work with the prediction module. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

All functions have their Docstring containing its main purpose and the needed parameters (physical description and formats). Please, do not hesitate to use it.

9.3.1 0. Index

1. *Predicting stellar occultation*
2. *Setting and/or modifying parameters*
3. *Apparent Sidereal Time*
4. *Ksi and Eta projection*
5. *MPC observatories*

9.3.2 1. Predicting stellar occultation

To predict a stellar occultation, the function `prediction()` must be used. It will need a `Body` Object and the time interval for the search. The function will generate the ephemeris within the given interval (with a default step of 60 seconds between positions, that may be changed by the user) and download the star coordinates from VizieR. The function is able to download stars from Gaia-DR2, Gaia-EDR3 and Gaia-DR3. The user must select the one desired by the keys *"gaiadr2"*, *"gaiaedr3"* or *"gaiadr3"* (default in v0.3). For every star, it checks the closest ephemeris. An occultation is identified if the distance is smaller than the radius of the Earth plus the radius of the object (given in `Body`) plus the error of the ephemeris (multiplied by a sigma factor given by the user). In the next step, the coordinate of the star is propagated to date using proper motion and parallax. It results in the occultation parameters, such as the Closest Approach distance, the Position Angle, Velocity, etc.

The function returns a `PredictionTable` with all the information.

[1]:
```
from sora.prediction import prediction
```

```
SORA version: 0.3
```

[2]:
```
prediction?
```

```
Signature:
prediction(
    time_beg,
    time_end,
    body=None,
    ephem=None,
    mag_lim=None,
    catalogue='gaiadr3',
    step=60,
    divs=1,
    sigma=1,
    radius=None,
    verbose=True,
    reference_center='geocenter',
)
Docstring:
Predicts stellar occultations.

Parameters
----------
time_beg : `str`, `astropy.time.Time`, required
    Initial time for prediction.

time_end : `str`, `astropy.time.Time`, required
    Final time for prediction.

body : `sora.Body`, `str`, default=None
    Object that will occult the stars. It must be a Body object or its name
    to search in the Small Body Database.

ephem : `sora.Ephem`, default=None
    object ephemeris. It must be an Ephemeris object.
    If using a EphemHorizons object, please use 'divs' to make division
    at most a month, or a timeout error may be raised by the Horizon query.
```

(continues on next page)

(continued from previous page)

```
mag_lim : `int`, `float`, `dict`, default=None
    Faintest magnitude allowed in the search. If the catalogue has more
    than one band defined in the catalogue object, the magnitude limit can
    be done for a specific band or a set of band. Ex: ``mag_lim={'V': 15}``,
    which will only download stars with V<=15 or ``mag_lim={'V': 15, 'B': 14}``
    which will download stars with V<=15 AND B<=14.

catalogue : `str`, `VizierCatalogue`
    The catalogue to download data. It can be ``'gaiadr2'``, ``'gaiaedr3'``,
    ``'gaiadr3'``, or a VizierCatalogue object. default='gaiadr3'

step : `int`, `float`, default=60
    Step, in seconds, of ephem times for search

divs : `int`, default=1
    Number of regions the ephemeris will be split for better search of
    occultations.

sigma : `int`, `float`, default=1
    Ephemeris error sigma for search off-Earth.

radius : `int`, `float`, default=None
    The radius of the body. It is important if not defined in body or ephem.

verbose : `bool`, default=True
    To show what is being done at the moment.

reference_center : `str`, `sora.Observer`, `sora.Spacecraft`
    A SORA observer object or a string 'geocenter'.
    The occultation parameters will be calculated in respect
    to this reference as center of projection. If a Spacecraft
    is used, please use smaller step since the search will be based
    on the target size and ephemeris error only.


Important
---------
When instantiating with "body" and "ephem", the user may call the function
in 3 ways:

1 - With "body" and "ephem".

2 - With only "body". In this case, the "body" parameter must be a Body
object and have an ephemeris associated (see Body documentation).

3 - With only "ephem". In this case, the "ephem" parameter must be one of
the Ephem Classes and have a name (see Ephem documentation) to search
for the body in the Small Body Database.

Returns
-------
 : `sora.prediction.PredictionTable`
```

(continues on next page)

(continued from previous page)

```
    PredictionTable with the occultation params for each event.
File:       ~/Documentos/códigos/SORA/sora/prediction/core.py
Type:       function
```

Importing and defining the `Body` to be used.

As showed in the docstring of the `prediction()` function, the user can give a `Body` object, an `Ephem` object, or both (given their restrictions). Please refer to the `Body` tutorial for further explanation.

Important Note: The files "de438_small.bsp" and "Chariklo.bsp" are modified files for the purpose of this guideline. If the user wants to predict for different object or epoch, proper files should be provided.

[3]:
```
from sora.body import Body
from sora.ephem import EphemKernel

chariklo = Body(name='Chariklo')
ephem = EphemKernel(spkid='2010199', kernels=['input/bsp/Chariklo.bsp', 'input/bsp/de438_
↪small.bsp'], error_ra=0.01, error_dec=0.01)
# here I'm using Body and Ephem so I'm able to give the errors in RA and DEC.
chariklo.ephem = ephem
```

```
Obtaining data for Chariklo from SBDB
```

[4]:
```
pred = prediction(body=chariklo, time_beg='2017-06-20 00:00:00', time_end='2017-07-01 00:
↪00:00', mag_lim=18, divs=3)
```

```
Ephemeris was split in 3 parts for better search of stars

Searching occultations in part 1/3
Generating Ephemeris between 2017-06-20 00:00:00.000 and 2017-06-23 15:59:00.000 ...
Downloading stars ...
    15 GaiaDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 2/3
Generating Ephemeris between 2017-06-23 16:00:00.000 and 2017-06-27 07:59:00.000 ...
Downloading stars ...
    11 GaiaDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 3/3
Generating Ephemeris between 2017-06-27 08:00:00.000 and 2017-06-30 23:59:00.000 ...
Downloading stars ...
    18 GaiaDR3 stars downloaded
Identifying occultations ...

8 occultations found.
```

Important

The default prediction is made considering the geocenter as center of reference. However, the prediction can also be made for a specific observer. In this case, it is important to note that the search considers only the size of the object and ephemeris error. If the object is very small, consider using a time `step` for initial prediction smaller, or using a different `radius` to avoid losing a promising event. Notice the occultation parameters (closest appoach time and distance, ...)

are referred to the observer.

[5]:
```
from sora import Observer
outeniqua = Observer(name='Outeniqua', lon='+16 49 17.710', lat='-21 17 58.170',␣
↪height=1416,
                     ephem=['input/bsp/Chariklo.bsp', 'input/bsp/de438_small.bsp'])

obs_pred = prediction(body=chariklo, time_beg='2017-06-20 00:00:00', time_end='2017-07-
↪01 00:00:00', step=10,
                      mag_lim=18, divs=3, reference_center=outeniqua)
```

```
Ephemeris was split in 3 parts for better search of stars

Searching occultations in part 1/3
Generating Ephemeris between 2017-06-20 00:00:00.000 and 2017-06-23 15:59:50.000 ...
Downloading stars ...
    10 GaiaDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 2/3
Generating Ephemeris between 2017-06-23 16:00:00.000 and 2017-06-27 07:59:50.000 ...
Downloading stars ...
    8 GaiaDR3 stars downloaded
Identifying occultations ...

Searching occultations in part 3/3
Generating Ephemeris between 2017-06-27 08:00:00.000 and 2017-06-30 23:59:50.000 ...
Downloading stars ...
    15 GaiaDR3 stars downloaded
Identifying occultations ...

1 occultations found.
```

[6]:
```
obs_pred[0]
```

[6]:
```
<PredictRow index=0>
         Epoch             ICRS Star Coord at Epoch      Geocentric Object Position    ␣
↪C/A     P/A     Vel     Dist     G      long   loct  M-G-T   S-G-T   GaiaDR3 Source ID
                                                                                        ␣
↪arcsec   deg    km / s    AU     mag     deg   hh:mm   deg     deg
        object                    object                          object           ␣
↪float64 float64 float64 float64 float64 float64  str5 float64 float64        str19
---------------------- ------------------------------ ------------------------------ ---
↪---- ------- ------- ------- ------- ------- ----- ------- ------- -------------------
2017-06-22 21:21:25.600 18 55 15.65251 -31 31 21.67062 18 55 15.65251 -31 31 21.67688  ␣
↪0.006  179.52  -22.36  14.659  14.224      52 00:50     149     166 6760223758801661440
```

[7]:
```
pred['2017-06-22 21']
```

[7]:
```
<PredictRow index=2>
         Epoch             ICRS Star Coord at Epoch      Geocentric Object Position    ␣
↪C/A     P/A     Vel     Dist     G      long   loct  M-G-T   S-G-T   GaiaDR3 Source ID
                                                                                        ␣
↪arcsec   deg    km / s    AU     mag     deg   hh:mm   deg     deg
```

(continues on next page)

(continued from previous page)

```
         object                       object                          object           ␣
↪float64 float64 float64 float64 float64 float64  str5 float64 float64        str19
---------------------- ------------------------------ ------------------------------ ---
↪---- ------- ------- ------- ------- ------- ----- ------- ------- -------------------
2017-06-22 21:18:48.200 18 55 15.65251 -31 31 21.67062 18 55 15.65249 -31 31 21.62190  ␣
↪0.049  359.72  -22.00  14.659  14.224      53 00:50     149     166 6760223758801661440
```

9.3.3 2. The PredictionTable

The resulting variable of the prediction function is a `PredictionTable` Object. The `PredictionTable` Class was created to have all the information about the occultations predicted, plot occultation maps and save the table in other formats.

The `PredictionTable` is a class that inherits all methods from `Astropy.table.Table`. All the functions present in `Astropy.table.Table` should work for `PredictionTable`.

More information about it at https://docs.astropy.org/en/stable/table/

if the user asks to print the prediction, a summary of the table is shown.

[8]:
```
print(pred)
```

```
         Epoch             ICRS Star Coord at Epoch    ...  GaiaDR3 Source ID
                                                       ...
---------------------- ------------------------------ ... --------------------
2017-06-21 09:57:43.440 18 55 36.17454 -31 31 19.03261 ... 6760228702284187264
2017-06-22 02:58:37.480 18 55 26.31652 -31 31 20.38440 ... 6760228839723992320
2017-06-22 21:18:48.200 18 55 15.65251 -31 31 21.67062 ... 6760223758801661440
2017-06-23 21:34:37.660 18 55 01.48119 -31 31 22.44257 ... 6760223513963694208
2017-06-24 10:13:39.980 18 54 54.06755 -31 31 22.36732 ... 6760226503261782656
2017-06-25 10:55:09.320 18 54 39.55296 -31 31 21.69987 ... 6760225163236852864
2017-06-26 08:14:35.540 18 54 26.97503 -31 31 20.51255 ... 6760226060885482624
2017-06-26 20:46:07.140 18 54 19.56971 -31 31 20.43382 ... 6760225712991422208
```

to print the complete table, we must use:

[9]:
```
pred.pprint_all()
```

```
         Epoch             ICRS Star Coord at Epoch      Geocentric Object Position    C/
↪A    P/A    Vel     Dist    G    long  loct M-G-T S-G-T  GaiaDR3 Source ID
                                                                                        ␣
↪arcsec  deg   km / s    AU    mag   deg  hh:mm  deg   deg
---------------------- ------------------------------ ------------------------------ ---
↪--- ------ ------ ------- ------ ---- ----- ----- ----- --------------------
2017-06-21 09:57:43.440 18 55 36.17454 -31 31 19.03261 18 55 36.17500 -31 31 19.60516  0.
↪573 179.41 -21.84  14.663 15.254  225 00:56   128   165 6760228702284187264
2017-06-22 02:58:37.480 18 55 26.31652 -31 31 20.38440 18 55 26.31674 -31 31 20.74213  0.
↪358 179.55 -21.92  14.661 17.929  329 00:53   138   166 6760228839723992320
2017-06-22 21:18:48.200 18 55 15.65251 -31 31 21.67062 18 55 15.65249 -31 31 21.62190  0.
↪049 359.72 -22.00  14.659 14.224   53 00:50   149   166 6760223758801661440
2017-06-23 21:34:37.660 18 55 01.48119 -31 31 22.44257 18 55 01.48116 -31 31 22.22700  0.
↪216 359.91 -22.10  14.657 17.818   48 00:46   162   167 6760223513963694208
2017-06-24 10:13:39.980 18 54 54.06755 -31 31 22.36732 18 54 54.06755 -31 31 22.28669  0.
```

(continues on next page)

(continued from previous page)

```
↪081   0.01 -22.15  14.656 17.928  218 00:43   167   168 67602265032617826 56
2017-06-25 10:55:09.320 18 54 39.55296 -31 31 21.69987 18 54 39.55291 -31 31 21.89148  0.
↪192 180.23 -22.23  14.654 16.955  206 00:39   164   168 6760225163236852864
2017-06-26 08:14:35.540 18 54 26.97503 -31 31 20.51255 18 54 26.97476 -31 31 20.99937  0.
↪487 180.41 -22.30  14.653 17.603  245 00:35   154   169 6760226060885482624
2017-06-26 20:46:07.140 18 54 19.56971 -31 31 20.43382 18 54 19.56985 -31 31 20.23484  0.
↪199   0.51 -22.33  14.652 17.702   57 00:33   147   169 6760225712991422208
```

M-G-T stands for Moon-Geocenter-Target, the on-sky angle between the star and the moon

S-G-T stands for Sun-Geocenter-Target, the on-sky angle between the star and the sun

The `PredictionTable` can be exported to PRAIA format file

In this case, some information, such as the Gaia-DR2 Source ID, S-G-T and M-G-T, are lost.

[10]:
```
pred.to_praia?
```

```
Signature: pred.to_praia(filename)
Docstring:
Writes PredictionTable to PRAIA format.

Parameters
----------
filename : `str`
    Name of the file to save table.
File:      ~/Documentos/códigos/SORA/sora/prediction/table.py
Type:      method
```

[11]:
```
pred.to_praia('output/Chariklo_occs.table')
# Check your folder named output
```

The `PredictionTable` can also generate the files in the Occult Watcher feed format

These are two files, named 'LOG.dat' and 'tableOccult_update.txt'. The occult watcher designation for the object must be informed. For instance, for the satellite Himalia, the designation is "P5M06". If `mode='append'` the prediction will be appended to an existing file. If `mode='restart'`, the file will be overwritted.

[12]:
```
pred.to_ow?
```

```
Signature: pred.to_ow(ow_des, mode='append')
Docstring:
Writes PredictionTable to OccultWatcher feeder update file format.
Tables will be saved in two files: "tableOccult_update.txt" and "LOG.dat"

Parameters
----------
ow_des : `str`
    Occult Watcher designation for the object.

mode : `str`, default='append'
    Use ``'append'`` to append table to already existing file and
    ``'restart'`` to overwrite existing file.
File:      ~/Documentos/códigos/SORA/sora/prediction/table.py
```

(continues on next page)

(continued from previous page)

```
Type:      method
```

[13]:
```
pred.to_ow('1997CU26')
```

Using the functions from `Astropy.table.Table` , the data can be exported to many other formats using the funtion 'write'

The formats allowed can be seen in https://docs.astropy.org/en/stable/io/unified.html#built-in-readers-writers. This includes latex and csv.

[14]:
```
pred.write('output/prediction.tex', overwrite=True)
# Check your folder named output
```

9.3.4 3. Using PredictionTable

The `PredictionTable` was not created to be instantiated directly by the user. It was designed to be an insteresting output object from the prediction function. However, a `PredictionTable` can be instantiated from a PRAIA occultation table. For this, a `PredictionTable` method must be called directly as shown below.

[15]:
```
from sora.prediction import PredictionTable
```

[16]:
```
PredictionTable.from_praia?
```

```
Signature: PredictionTable.from_praia(filename, name, **kwargs)
Docstring:
Creates a PredictionTable Table reading from a PRAIA table.

Parameters
----------
filename : `str`
    Path to the PRAIA table file.

name : `str`
    Name of the Object of the prediction.

radius : `int`, `float`, optional
    Object radius, in km.
    If not given it's searched in online database.
    When not found online, the default is set to zero.

Returns
-------
 : `sora.prediction.PredictionTable`
    A PredictionTable object.
File:      ~/Documentos/códigos/SORA/sora/prediction/table.py
Type:      method
```

[17]:
```
pred_1 = PredictionTable.from_praia(filename='output/Chariklo_occs.table', name='Chariklo
↪')
```

```
Obtaining data for chariklo from SBDB
```

[18]:
```
pred_1.pprint_all()
```

```
         Epoch             ICRS Star Coord at Epoch      Geocentric Object Position    C/
↪A    P/A    Vel     Dist    G    long  loct M-G-T S-G-T  Source ID
                                                                                        ␣
↪arcsec  deg   km / s    AU    mag   deg  hh:mm  deg   deg
----------------------- ------------------------------- ------------------------------ ---
↪--- ------ ------ ------- ------ ---- ----- ----- ----- -----------
2017-06-21 09:57:43.400 18 55 36.17450 -31 31 19.03260 18 55 36.17500 -31 31 19.60520  0.
↪573 179.41 -21.84  14.660 15.204  225 00:56   128   165
2017-06-22 02:58:37.400 18 55 26.31650 -31 31 20.38440 18 55 26.31670 -31 31 20.74210  0.
↪358 179.55 -21.92  14.660 17.900  329 00:53   138   166
2017-06-22 21:18:48.200 18 55 15.65250 -31 31 21.67060 18 55 15.65250 -31 31 21.62190  0.
↪049 359.72 -22.00  14.660 14.197   53 00:50   149   166
2017-06-23 21:34:37.600 18 55 01.48120 -31 31 22.44260 18 55 01.48120 -31 31 22.22700  0.
↪216 359.91 -22.10  14.660 17.792   48 00:46   162   167
2017-06-24 10:13:39.900 18 54 54.06750 -31 31 22.36730 18 54 54.06750 -31 31 22.28670  0.
↪081   0.01 -22.15  14.660 17.889  218 00:43   167   168
2017-06-25 10:55:09.300 18 54 39.55300 -31 31 21.69990 18 54 39.55290 -31 31 21.89150  0.
↪192 180.23 -22.23  14.650 16.985  206 00:39   164   168
2017-06-26 08:14:35.500 18 54 26.97500 -31 31 20.51260 18 54 26.97480 -31 31 20.99940  0.
↪487 180.41 -22.30  14.650 17.582  245 00:35   154   169
2017-06-26 20:46:07.100 18 54 19.56970 -31 31 20.43380 18 54 19.56980 -31 31 20.23480  0.
↪199   0.51 -22.33  14.650 17.680   57 00:33   147   169
```

To remove some occultations the User can use two functions. The first is the *PredictionTable.remove_occ()*.

[19]:
```
pred_1.remove_occ?
```

```
Signature: pred_1.remove_occ(date)
Docstring:
Removes stellar occultations from table.

Parameters
----------
date : `str`, `list`
    Date or list of dates of the occultation to be removed.
    The dates mut be as shown in the 'Epoch' column. If the date is not
    complete, the function will select all occultations that matches the
    given string. For instance, ``date='2020-06'`` will remove all
    occultations from the month of June 2020.
File:      ~/Documentos/códigos/SORA/sora/prediction/table.py
Type:      method
```

[20]:
```
pred_1.remove_occ('2017-06-21 09:57')
```

[21]:
```
pred_1.pprint_all()
```

```
         Epoch             ICRS Star Coord at Epoch      Geocentric Object Position    C/
↪A    P/A    Vel     Dist    G    long  loct M-G-T S-G-T  Source ID
                                                                                        ␣
```

(continues on next page)

(continued from previous page)

```
↪arcsec  deg   km / s    AU    mag   deg  hh:mm  deg   deg
---------------------- ------------------------------ ------------------------------ ---
↪--- ------ ------ ------- ------ ---- ----- ----- ----- ----------
2017-06-22 02:58:37.400 18 55 26.31650 -31 31 20.38440 18 55 26.31670 -31 31 20.74210  0.
↪358 179.55 -21.92  14.660 17.900  329 00:53   138   166
2017-06-22 21:18:48.200 18 55 15.65250 -31 31 21.67060 18 55 15.65250 -31 31 21.62190  0.
↪049 359.72 -22.00  14.660 14.197   53 00:50   149   166
2017-06-23 21:34:37.600 18 55 01.48120 -31 31 22.44260 18 55 01.48120 -31 31 22.22700  0.
↪216 359.91 -22.10  14.660 17.792   48 00:46   162   167
2017-06-24 10:13:39.900 18 54 54.06750 -31 31 22.36730 18 54 54.06750 -31 31 22.28670  0.
↪081   0.01 -22.15  14.660 17.889  218 00:43   167   168
2017-06-25 10:55:09.300 18 54 39.55300 -31 31 21.69990 18 54 39.55290 -31 31 21.89150  0.
↪192 180.23 -22.23  14.650 16.985  206 00:39   164   168
2017-06-26 08:14:35.500 18 54 26.97500 -31 31 20.51260 18 54 26.97480 -31 31 20.99940  0.
↪487 180.41 -22.30  14.650 17.582  245 00:35   154   169
2017-06-26 20:46:07.100 18 54 19.56970 -31 31 20.43380 18 54 19.56980 -31 31 20.23480  0.
↪199   0.51 -22.33  14.650 17.680   57 00:33   147   169
```

The second function is the `PredictionTable.keep_from_selected_images()` this function will check the names in the saved images and eliminate the occultations without the map. This functions allows that after the user deleted the maps of the unwanted occultation it will eliminate the respective rows in the `PredictionTable`.

[22]:
```
pred_1.keep_from_selected_images?
```

```
Signature: pred_1.keep_from_selected_images(path='.')
Docstring:
Keeps predictions which images were not deleted in given path.
This function uses the name of the images to identify predictions.
The name must be the automatic one generated by plot_occ_map().
The format of the image is not relevant.

Parameters
----------
path : `str`
    Path where images are located.
File:      ~/Documentos/códigos/SORA/sora/prediction/table.py
Type:      method
```

9.3.5 4. Plotting occultation maps

SORA is also able to generate an occultation map. The only required parameters are the occultation parameters. The function also has many inputs to configure the plot.

The first time the function is called, cartopy will download some data referring to the features presented in the map, such as the country and state division, lakes, rivers, etc.

[23]:
```
from sora.prediction import plot_occ_map
```

[24]:
```
plot_occ_map?
```

```
Signature:
plot_occ_map(
    name,
    radius,
    coord,
    time,
    ca,
    pa,
    vel,
    dist,
    mag=0,
    longi=0,
    **kwargs,
)
Docstring:
Plots the map of the occultation.

Parameters
----------
name : `str`
    Name of the object.

radius : `int`, `float`
    Radius of the object, in km.

coord : `str`, `astropy.coordinates.SkyCoord`
    Coordinates of the star (``"hh mm ss.sss dd mm ss.sss"`` or
    ``"hh.hhhhhhhh dd.dddddddd"``).

time : `str`, `astropy.time.Time`
    Instant of Closest Approach (iso or isot format).

ca : `int`, `float`
    Closest Approach Distance, in arcsec.

pa : `int`, `float`
    Position Angle at C/A, in degrees.

vel : `int`, `float`
    Velocity of the event, in km/s.

dist : `int`, `float`
    Object distance at C/A, in AU.

mag : `int`, `float`, default=0
    Mag* = Normalized magnitude to vel=20km/s.

longi : `int`, `float`, default=0
    East longitude of sub-planet point, deg, positive towards East.

nameimg : `str`
    Change the name of the imaged saved.
```

(continues on next page)

(continued from previous page)

```
path : `str`
    Path to a directory where to save map.

resolution : `int`, default=2
    Cartopy feature resolution.

    - ``1`` means a resolution of "10m";

    - ``2`` a resolution of "50m";

    - ``3`` a resolution of "100m".

states : `bool`
    If True, plots the states borders of the countries. The states
    of some countries will only be shown depending on the resolution.

zoom : `int`, `float`
    Zooms in or out of the map.

centermap_geo : `list`, default=None
    Center the map given coordinates in longitude and latitude. It must be
    a list with two numbers.

centermap_delta : `list`, default=None
    Displace the center of the map given displacement in X and Y, in km.
    It must be a list with two numbers.

centerproj : `list`
    Rotates the Earth to show occultation with the center projected at a
    given longitude and latitude. It must be a list with two numbers.

labels : `bool`, default=True
    Plots text above and below the map with the occultation parameters.

meridians : `int`, default=30
    Plots lines representing the meridians for given interval, in degrees.

parallels : `int`, default=30
    Plots lines representing the parallels for given interval, in degrees.

sites : `dict`
    Plots site positions in map. It must be a python dictionary where the
    key is  the `name` of the site, and the value is a list with `longitude`,
    `latitude`, `delta_x`, `delta_y` and `color`. `delta_x` and `delta_y`
    are displacement, in km, from the point position of the site in the map
    and the `name`. `color` is the color of the point.

site_name : `bool`
    If True, it prints the name of the sites given, else it plots only the points.

site_box_alpha : `int`, `float`, default=0
    Sets the transparency of a box surrounding each station name. 0 equals to
```

(continues on next page)

(continued from previous page)

```
    transparent, and 1 equals to opaque.

countries : `dict`
    Plots the names of countries. It must be a python dictionary where the
    key is the name of the country and the value is a list with longitude
    and latitude of the lower left part of the text.

offset : `list`
    Applies an offset to the ephemeris, calculating new CA and instant of
    CA. It is a pair of `delta_RA*cosDEC` and `delta_DEC`.

mapstyle : `int`, default=1
    Define the color style of the map. ``'1''' is the default black
    and white scale. ``'2''' is a colored map.

error : `int`, `float`
    Ephemeris error in mas. It plots a dashed line representing radius + error.

ercolor : `str`
    Changes the color of the lines of the error bar.

ring : `int`, `float`
    Plots a dashed line representing the location of a ring. It is given
    in km, from the center.

rncolor : `str`
    Changes the color of ring lines.

atm : `int`, `float`
    Plots a dashed line representing the location of an atmosphere. It is
    given in km, from the center.

atcolor : `str`
    Changes the color of atm lines.

chord_delta : `list`
    List with distances from center to plot chords.

chord_geo : `2d-list`
    List with pairs of coordinates to plot chords.

chcolor : `str`, default='grey'
    Color of the line of the chords.

heights : `list`
    It plots a circular dashed line showing the locations where the observer
    would observe the occultation at a given height above the horizons.
    This must be a list.

hcolor : `str`
    Changes the color of the height lines.
```

(continues on next page)

(continued from previous page)

```
mapsize : `list`, default= [46.0, 38.0]
    The size of figure, in cm. It must be a list with two values.

cpoints : `int`, `float`, default=60
    Interval for the small points marking the center of shadow, in seconds.

ptcolor : `str`
    Change the color of the center points.

alpha : `float`, default=0.2
    The transparency of the night shade, where 0.0 is full transparency and
    1.0 is full black.

fmt : `str`, default:'png'
    The format to save the image. It is parsed directly by `matplotlib.pyplot`.

dpi : `int`, default=100
    Resolution in "dots per inch". It defines the quality of the image.

lncolor : `str`
    Changes the color of the line that represents the limits of the shadow
    over Earth.

outcolor :`str`
    Changes the color of the lines that represents the limits of the shadow
    outside Earth.

scale : `int`, `float`
    Arbitrary scale for the size of the name of the site.

cscale : `int`, `float`
    Arbitrary scale for the name of the country.

sscale : `int`, `float`
    Arbitrary scale for the size of point of the site.

pscale : `int`, `float`
    Arbitrary scale for the size of the points that represent the center of
    the shadow.

arrow : `bool`
    If True, it plots the arrow with the occultation direction.


Important
---------
Required parameters to plot an occultation map: 'name', 'radius', 'coord',
'time', 'ca', 'pa', 'vel', and 'dist'.


Note
----
```

(continues on next page)

(continued from previous page)

```
The parameters 'mag' and 'longi' are optional and only printed in label.
All other remaining parameters can be used to further customize the Map
configuration.

When producing the map, only one of 'centermap_geo' or 'centermap_delta'
options can be used at a time.
File:      ~/Documentos/códigos/SORA/sora/prediction/occmap.py
Type:      function
```

To make it easier to plot the predictions, the ``PredictionTable.plot_occ_map()`` function can be called which automatically fills the required parameters.

To plot the map for all the predictions

[25]:
```
pred.plot_occ_map(path='output/')
```

```
10199 Chariklo_2017-06-21T09_57_43.440.png generated
10199 Chariklo_2017-06-22T02_58_37.480.png generated
10199 Chariklo_2017-06-22T21_18_48.200.png generated
10199 Chariklo_2017-06-23T21_34_37.660.png generated
10199 Chariklo_2017-06-24T10_13_39.980.png generated
10199 Chariklo_2017-06-25T10_55_09.320.png generated
10199 Chariklo_2017-06-26T08_14_35.540.png generated
10199 Chariklo_2017-06-26T20_46_07.140.png generated
```

To plot the map for only one prediction, just give an item to the `PredictionTable` object.

[26]:
```
pred[0].plot_occ_map(path='output/')
```

```
10199 Chariklo_2017-06-21T09_57_43.440.png generated
```

The `PredictionTable` can also be plotted by giving the date of the occultation. All the occultations that matches the date will be plotted.

The date can be as constrained as the user wants, and must match the text that appears in the *'Epoch'* column.

[27]:
```
pred['2017-06-26'].plot_occ_map()
```

```
10199 Chariklo_2017-06-26T08_14_35.540.png generated
10199 Chariklo_2017-06-26T20_46_07.140.png generated
```

9.3.6 5. Occultation parameters

Finally, a function is implemented in the prediction module which calculates the occultation parameters. For this function, it must be passed a `Star` Object, an `Ephem` Object (it can be any of the `Ephem` classes) and a time. The time does not need to be precise, but it must be close within 60 minutes from the Closest Approach time.

[28]:
```
from sora.prediction import occ_params
```

[29]:
```
occ_params?
```

```
Signature:
occ_params(
```

(continues on next page)

(continued from previous page)

```
    star,
    ephem,
    time,
    n_recursions=5,
    max_tdiff=None,
    reference_center='geocenter',
)
Docstring:
Calculates the parameters of the occultation, as instant, CA, PA.

Parameters
----------
star : `sora.Star`
    The coordinate of the star in the same reference frame as the ephemeris.
    It must be a Star object.

ephem : `sora.Ephem*`
    Object ephemeris. It must be an Ephemeris object.

time : `astropy.time.Time`
    Time close to occultation epoch to calculate occultation parameters.

n_recursions : `int`, default=5
    The number of attempts to try obtain prediction parameters in case the
    event is outside the previous range of time.

max_tdiff : `int`, default=None
    Maximum difference from given time it will attempt to identify the
    occultation, in minutes. If given, 'n_recursions' is ignored.

reference_center : `str`, `sora.Observer`, `sora.Spacecraft`
        A SORA observer object or a string 'geocenter'.
        The occultation parameters will be calculated in respect
        to this reference as center of projection.

Returns
-------
 Oredered list : `list`
    - Instant of CA (Time): Instant of Closest Approach.

    - CA (arcsec): Distance of Closest Approach.

    - PA (deg): Position Angle at Closest Approach.

    - vel (km/s): Velocity of the occultation.

    - dist (AU): the object geocentric distance.
File:      ~/Documentos/códigos/SORA/sora/prediction/core.py
Type:      function
```

[30]:
```
from sora.star import Star

s = Star(code='6760225712991422208', verbose=False)
```

[31]:
```
tca, ca, pa, vel, dist = occ_params(star=s, ephem=chariklo.ephem, time='2017-06-26 20:40
↪')

print('Time of the CA:   {} UTC'.format(tca))
print('Closest Approach: {:.3f}'.format(ca))
print('Position Angle:   {:.3f}'.format(pa))
print('Shadow Velocity:  {:.3f}'.format(vel))
print('Object Distance:  {:.5f}'.format(dist))
```

```
Time of the CA:   2017-06-26 20:46:07.140 UTC
Closest Approach: 0.199 arcsec
Position Angle:   0.511 deg
Shadow Velocity:  -22.333 km / s
Object Distance:  14.65229 AU
```

This Jupyter-Notebook was designed as a tutorial for how to work with the `prediction()` function and `PredictionTable` Class. More information about the other classes, please refer to their specif Jupyter-Notebook. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The End

9.4 Observer and Spacecraft Classes

The `Observer` and `Spacecraft` Classes within SORA were created to deal with the observer location. The online documentation here contains the details about every step.

This Jupyter-Notebook was designed as a tutorial for how to work with the `Observer` and `Spacecraft` Classes. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The `Observer` and `Spacecraft` Docstring were designed to help the users. Also, each function has its Docstring containing its main purpose and the needed parameters (physical description and formats). Please, do not hesitate to use it.

9.4.1 0. Index

1. *Instantiating an Observer or Spacecraft Object*
2. *Setting and/or modifing parameters*
3. *Apparent Sidereal Time*
4. *Ksi and Eta projection*
5. *MPC observatories*

[1]:
```
## import the Class
from sora.observer import Observer, Spacecraft
## This means the Observer Class is imported from the "observer" module of the sora␣
```

(continues on next page)

(continued from previous page)

```
↪package

## To facilitate, sora allows to import Observer directly from the sora package.
from sora import Observer, Spacecraft
```

```
SORA version: 0.3
```

9.4.2 1. Instantiating an Observer or Spacecraft Object

1.1 Observer

The `Observer` Object Class can be instantiated in different ways. First, all observers should have a *name* that will distinguish it from other `Observer` Objects and also allows the `Occultation` Object to control all the steps and Objects. This is the name which the user will use to refer to this observer within the `Occultation` Object. The name is not needed when the site is downloaded from the MPC database in which case the name in the MPC will be used.

The `Observer` main objective is to deal with the observer location, convert the coordinates from ITRS and GCRS and vice-versa and calculate the Orthographic projection the coordinates in the direction of a coordinate.

[2]:
```
Observer?
```

```
Init signature: Observer(**kwargs)
Docstring:
Defines the observer object.

Attributes
----------
name : `str`
    Name for the Observer. Observer is uniquely defined (name must be
    different for each observer).

code : `str`
    The IAU code for SORA to search for its coordinates in MPC database.

site : `astropy.coordinates.EarthLocation`
    User provides an EarthLocation object.

lon : `str`, `float`
    The Longitude of the site in degrees.
    Positive to East. Range (0 to 360) or (-180 to +180).
    User can provide in degrees (`float`) or hexadecimal (`string`).

lat : `str`, `float`
    The Latitude of the site in degrees.
    Positive North. Range (+90 to -90).
    User can provide in degrees (float) or hexadecimal (string).

height : `int`, `float`
    The height of the site in meters above see level.

ephem : `str`, `list`
    The ephemeris used to locate the observer on space.
```

(continues on next page)

(continued from previous page)

```
    It can be "horizons" to use horizons or a list of kernels


Examples
--------
User can provide one of the following to define an observer:

- If user will use the MPC name for the site:

>>> Observer(code)

- If user wants to use a different name from the MPC database:

>>> Observer(name, code)

- If user wants to use an EarthLocation value:

>>> from astropy.coordinates import EarthLocation
>>> EarthLocation(lon, lat, height)
>>> Observer(name, site)

- If user wants to give site coordinates directly:

>>> Observer(name, lon, lat, height)
File:           ~/Documentos/códigos/SORA/sora/observer/core.py
Type:           type
Subclasses:
```

Example of an `Observer` intantiated with a MPC code

[3]:
```
opd = Observer(code='874')
```

[4]:
```
## To see the name of the object "opd"
opd.name
```

[4]:
```
'Observatorio do Pico dos Dias, Itajuba'
```

In this case, the name is a bit too long, the user can define a differente name upon instantiation.

[5]:
```
opd = Observer(name='OPD', code='874')
```

Example of `Observer` instantiated with coordinates

This coordinates must in degrees for longitude and latitude and in meters in height.

[6]:
```
# using string representation
obs1 = Observer(name='Ex 1', lon='-45 34 57', lat='-22 32 04', height=1864)
```

[7]:
```
# using numeric representation
obs2 = Observer(name='Ex 2', lon=-69.295805, lat=-31.79877, height=2495)
```

Example using an Astropy `EarthLocation` Object

Astropy has an `EarthLocation` Class with observer location information. SORA uses this class as a core functionality in the `Observer` Class. So the user can give an object created from `EarthLocation` as input.

[8]:
```
from astropy.coordinates import EarthLocation
site = EarthLocation(lon='-45 34 57', lat=' -22 32 04', height=1864)
obs3 = Observer(name='Ex 3', site=site)
```

The user can access some information from the object as attribute

[9]:
```
obs3.name
```

[9]:
```
'Ex 3'
```

[10]:
```
obs3.lon
```

[10]: $-45^\circ 34' 57''$

[11]:
```
obs3.lat
```

[11]: $-22^\circ 32' 04''$

[12]:
```
obs3.height
```

[12]: $1864\ \mathrm{m}$

If the user wants to see the informations in the object, just prints the object directly.

[13]:
```
print(obs3)
```

```
Site: Ex 3
Geodetic coordinates: Lon: -45d34m57s, Lat: -22d32m04s, height: 1.864 km
```

1.2 Spacecraft

In SORA, it is possible to inform an spacecraft observation. The `Spacecraft` Class provides functionality to identify the observer in space. This way, the `Occultation` object can calculate the geometry for the combination of different `Sacecraft` and `Observer` objects.

For this, to instantiate a `Spacecraft` object, it is required an ephemeris. The options available are: the string "horizons" to use ephemeris from the Horizons web service; a list of spice kernels where Spice can identify the spacecraft relative to the barycenter of the Solar System. A "spkid" is also required. **Please do not use an Ephem object in this case.**

[14]:
```
cassini = Spacecraft(name='Cassini', spkid='-82', ephem='horizons')
```

[15]:
```
# Using kernels. For example, downloading the New Horizons kernel from
# https://naif.jpl.nasa.gov/pub/naif/pds/data/nh-j_p_ss-spice-6-v1.0/nhsp_1000/data/spk/

# new_horizons = Spacecraft(name='New Horizons', spkid='-98', ephem=['nh_recon_pluto_od122_
↪v01.bsp'])
```

9.4.3 2. Setting and/or modifing parameters

If any parameter given to `Observer` is not correct, the user does not need to delete or overwrite the previous object. It can be given the new values directly to the object attributes. Only the *name* attribute cannot be replaced.

[16]:
```
print(obs3.lon)
```
```
-45d34m57s
```

[17]:
```
obs3.lon = '-35 40 26'
print(obs3.lon)
```
```
-35d40m26s
```

[18]:
```
obs3.lon = 67.3514
print(obs3.lon)
```
```
67d21m05.04s
```

[19]:
```
obs3.name = 'Teste'
```
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [19], in <cell line: 1>()
----> 1 obs3.name = 'Teste'

AttributeError: can't set attribute
```

9.4.4 3. Apparent Sidereal Time

If the user, for some reason, wants to know the apparent sidereal time at a certain location for a given time, there exists a function that calculates it. This function is only available for the `Observer` object.

[20]:
```
obs3.sidereal_time?
```
```
Signature: obs3.sidereal_time(time, mode='local')
Docstring:
Calculates the Apparent Sidereal Time at a reference time.

Parameters
----------
time : `str`, `astropy.time.Time`
    Reference time to calculate sidereal time.It can be a string
    in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

mode : `str`
    Local or greenwich time.
    If mode set ``'local'`` calculates the sidereal time for the
    coordinates of this object.
    If mode set ``'greenwich'`` calculates the Greenwich Apparent
    Sidereal Time.


Returns
```

(continues on next page)

(continued from previous page)

```
-------
sidereal_time
    An Astropy Longitude object with the Sidereal Time.
File:      ~/Documentos/códigos/SORA/sora/observer/core.py
Type:      method
```

[21]:
```
obs3.sidereal_time(time='2020-05-10 00:00:00')
```

[21]: $19^{\mathrm{h}}42^{\mathrm{m}}24.39237631^{\mathrm{s}}$

The input time can also be an Astropy `Time` Object

[22]:
```
from astropy.time import Time

t = Time('2020-05-10 00:00:00')

obs3.sidereal_time(t)
```

[22]: $19^{\mathrm{h}}42^{\mathrm{m}}24.39237631^{\mathrm{s}}$

If the user wants to calculate the Greenwich Apparent Sidereal Time, it can be done as:

[23]:
```
obs3.sidereal_time('2020-05-10 00:00:00', mode='greenwich')
```

[23]: $15^{\mathrm{h}}13^{\mathrm{m}}00.05637697^{\mathrm{s}}$

Also, with the Observer the user can calculate the altitude and azimuth of a given target.

[24]:
```
obs3.altaz?
```

```
Signature: obs3.altaz(time, coord)
Docstring:
Calculates the Altitude and Azimuth at a reference time for a coordinate.

Parameters
----------
time : `str`, `astropy.time.Time`
    Reference time to calculate the sidereal time. It can be a string
    in the ISO format (yyyy-mm-dd hh:mm:ss.s) or an astropy Time object.

coord : `str`, `astropy.coordinates.SkyCoord`
    Coordinate of the target ICRS.


Returns
-------
altitude : `float`
    Object altitude in degrees.

azimuth : `float`
    Object azimuth in degrees.
File:      ~/Documentos/códigos/SORA/sora/observer/core.py
Type:      method
```

[25]:
```
altitude, azimuth = obs3.altaz(time='2020-05-10 00:00:00', coord='20 12 19.3 +25 30 00.5
↪')

print('Altitude {:.2f} degrees'.format(altitude))
print('Azimuth  {:.2f} degrees'.format(azimuth))
```

```
Altitude 41.33 degrees
Azimuth  9.25 degrees
```

9.4.5 4. Ksi and Eta projection

`Observer` object calculates the orthographic projection (ksi and eta) of a site in the direction of a star given the following function. Ksi is in the East direction and Eta in the North direction.

The coordinates of the star given must be in the Geocentric Celestial Reference System (GCRS).

This is calculated automaticaly in `Occultation`.

[26]:
```
obs3.get_ksi_eta?
```

```
Signature: obs3.get_ksi_eta(time, star)
Docstring:
Calculates relative position to star in the orthographic projection.

Parameters
----------
time : `str`, `astropy.time.Time`
    Reference time to calculate the position. It can be a string in the
    format ``'yyyy-mm-dd hh:mm:ss.s'`` or an astropy `Time` object.

star : `str`, `astropy.coordinates.SkyCoord`
    The coordinate of the star in the same reference frame as the
    ephemeris. It can be a string in the format ``'hh mm ss.s +dd mm ss.ss'``
    or an astropy `SkyCoord` object.


Returns
-------
ksi, eta : `float`
    On-sky orthographic projection of the observer relative to a star.
    ``ksi`` is in the North-South direction (North positive).
    ``eta`` is in the East-West direction (East positive).
File:      ~/Documentos/códigos/SORA/sora/observer/core.py
Type:      method
```

[27]:
```
ksi, eta = obs3.get_ksi_eta(time='2020-05-10 00:00:00', star='19 21 18.63201 -21 44 25.
↪3924')
print(ksi, eta)
```

```
511.2578070060709 -86.8256098119528
```

A list of times can be given to the time param, as shown below.

[28]:
```
times = ['2020-05-10 00:00:00', '2020-05-10 01:00:00', '2020-05-10 02:00:00', '2020-05-
↪10 03:00:00']
ksi, eta = obs3.get_ksi_eta(time=times, star='19 21 18.63201 -21 44 25.3924')
print(ksi)
print(eta)
```

```
[ 511.25780701 2017.74529995 3385.67428305 4521.3144453 ]
[ -86.82560981 -213.25930425 -479.77636105 -868.11505452]
```

9.4.6 5. MPC observatories

The user can download and look all the observatories from the MPC database. These are ground-based observatories. The spacecraft observatories are ignored. The function presented returns a Python dictionary that can be searched by the MPC code.

[29]:
```
from sora.observer import search_code_mpc
observatories = search_code_mpc()
```

[30]:
```
observatories['874']
```

[30]:
```
('Observatorio do Pico dos Dias, Itajuba',
 <EarthLocation (4126.27235742, -4211.05894345, -2429.98090308) km>)
```

[31]:
```
observatories['511']
```

[31]:
```
('Haute Provence',
 <EarthLocation (4578.31238503, 458.24339505, 4403.08309658) km>)
```

[32]:
```
observatories
```

[32]:
```
{'000': ('Greenwich', <EarthLocation (3980.65908307, 0., 4966.84662601) km>),
 '001': ('Crowborough',
  <EarthLocation (4017.70150872, 10.81285696, 4937.37963307) km>),
 '002': ('Rayleigh',
  <EarthLocation (3966.96894668, 42.92841063, 4981.324997) km>),
 '003': ('Montpellier',
  <EarthLocation (4613.44108454, 314.51286042, 4381.780119) km>),
 '004': ('Toulouse',
  <EarthLocation (4623.91819079, 118.053188, 4377.12407899) km>),
 '005': ('Meudon',
  <EarthLocation (4205.68487713, 163.84501614, 4776.42734587) km>),
 '006': ('Fabra Observatory, Barcelona',
  <EarthLocation (4786.95713426, 177.55185536, 4197.63692567) km>),
 '007': ('Paris',
  <EarthLocation (4202.69234465, 171.49762241, 4778.64693755) km>),
 '008': ('Algiers-Bouzareah',
  <EarthLocation (5106.30535438, 270.78274243, 3799.96646186) km>),
 '009': ('Berne-Uecht',
  <EarthLocation (4324.63496703, 564.87289002, 4638.1812264) km>),
 '010': ('Caussols',
  <EarthLocation (4581.93585771, 556.19867516, 4389.0193045) km>),
 '011': ('Wetzikon',
  <EarthLocation (4281.0645514, 662.55253581, 4666.30881057) km>),
```

(continues on next page)

(continued from previous page)

```
'012': ('Uccle',
 <EarthLocation (4027.8042438, 306.96764563, 4919.49533692) km>),
'013': ('Leiden',
 <EarthLocation (3909.35921921, 306.57274675, 5013.40064797) km>),
'014': ('Marseilles',
 <EarthLocation (4628.16865667, 437.09059074, 4352.33863861) km>),
'015': ('Utrecht',
 <EarthLocation (3911.73785333, 351.12903035, 5008.65531404) km>),
'016': ('Besancon',
 <EarthLocation (4313.83907875, 452.58826347, 4660.88739412) km>),
'017': ('Hoher List',
 <EarthLocation (4065.19923893, 488.28933342, 4874.69530263) km>),
'018': ('Dusseldorf-Bilk',
 <EarthLocation (3976.28374918, 471.41237353, 4948.03112186) km>),
'019': ('Neuchatel',
 <EarthLocation (4326.15188596, 527.92758915, 4641.94432723) km>),
'020': ('Nice',
 <EarthLocation (4579.76810312, 586.71432225, 4386.05347079) km>),
'021': ('Karlsruhe',
 <EarthLocation (4145.70029238, 611.11175641, 4792.40457906) km>),
'022': ('Pino Torinese',
 <EarthLocation (4473.57798878, 610.79957545, 4490.78248033) km>),
'023': ('Wiesbaden',
 <EarthLocation (4058.50935478, 589.35992305, 4868.75087895) km>),
'024': ('Heidelberg-Konigstuhl',
 <EarthLocation (4111.15268424, 630.68101342, 4819.9581309) km>),
'025': ('Stuttgart',
 <EarthLocation (4156.75143863, 672.98624573, 4774.90934927) km>),
'026': ('Berne-Zimmerwald',
 <EarthLocation (4331.25910881, 567.53807977, 4633.09147307) km>),
'027': ('Milan',
 <EarthLocation (4423.36528553, 715.73166154, 4523.94879273) km>),
'028': ('Wurzburg',
 <EarthLocation (4063.87610771, 711.91332679, 4847.95815233) km>),
'029': ('Hamburg-Bergedorf',
 <EarthLocation (3743.32398264, 676.26904867, 5102.5096) km>),
'030': ('Arcetri Observatory, Florence',
 <EarthLocation (4526.05681305, 900.65493691, 4388.23479364) km>),
'031': ('Sonneberg',
 <EarthLocation (3998.53165629, 790.99449901, 4890.14952858) km>),
'032': ('Jena',
 <EarthLocation (3946.54245519, 808.88500632, 4928.42472872) km>),
'033': ('Karl Schwarzschild Observatory, Tautenburg',
 <EarthLocation (3940.19976005, 816.7822946, 4932.42382062) km>),
'034': ('Monte Mario Observatory, Rome',
 <EarthLocation (4641.02540599, 1024.85124705, 4239.26702587) km>),
'035': ('Copenhagen',
 <EarthLocation (3517.23978013, 784.64431695, 5244.87599598) km>),
'036': ('Castel Gandolfo',
 <EarthLocation (4650.34562297, 1043.77129078, 4225.00551154) km>),
'037': ('Collurania Observatory, Teramo',
 <EarthLocation (4563.82215392, 1115.35040972, 4299.88483992) km>),
```

(continues on next page)

(continued from previous page)

```
'038': ('Trieste',
 <EarthLocation (4338.41313433, 1063.24025378, 4537.66178728) km>),
'039': ('Lund',
 <EarthLocation (3507.68447539, 821.90619231, 5245.57121291) km>),
'040': ('Lohrmann Institute, Dresden',
 <EarthLocation (3904.58595502, 953.98721689, 4935.84888019) km>),
'041': ('Innsbruck',
 <EarthLocation (4250.99032698, 855.66998387, 4662.49468464) km>),
'042': ('Potsdam',
 <EarthLocation (3800.65449726, 881.94260846, 5028.77195814) km>),
'043': ('Asiago Astrophysical Observatory, Padua',
 <EarthLocation (4360.00589905, 889.14790229, 4555.70553685) km>),
'044': ('Capodimonte Observatory, Naples',
 <EarthLocation (4681.91550472, 1189.56828909, 4151.01912234) km>),
'045': ('Vienna (since 1879)',
 <EarthLocation (4084.79368039, 1197.49604984, 4734.29975099) km>),
'046': ('Klet Observatory, Ceske Budejovice',
 <EarthLocation (4074.53476663, 1037.68469599, 4781.37040205) km>),
'047': ('Poznan',
 <EarthLocation (3731.98077349, 1132.31160776, 5030.05396368) km>),
'048': ('Hradec Kralove',
 <EarthLocation (3937.47507359, 1117.22211523, 4875.65202318) km>),
'049': ('Uppsala-Kvistaberg',
 <EarthLocation (3093.17587742, 981.61130552, 5472.441546) km>),
'050': ('Stockholm (before 1931)',
 <EarthLocation (3099.77689719, 1010.66083147, 5463.5121542) km>),
'051': ('Royal Observatory, Cape of Good Hope',
 <EarthLocation (5024.29945965, 1678.8257102, -3540.37628596) km>),
'052': ('Stockholm-Saltsjobaden',
 <EarthLocation (3101.75436836, 1026.30567636, 5459.49392789) km>),
'053': ('Konkoly Observatory, Budapest (since 1934)',
 <EarthLocation (4082.90178119, 1403.00286467, 4679.83046101) km>),
'054': ('Brorfelde',
 <EarthLocation (3535.14847688, 729.86795946, 5240.85139153) km>),
'055': ('Cracow',
 <EarthLocation (3856.05999997, 1400.4127008, 4867.53903292) km>),
'056': ('Skalnate Pleso',
 <EarthLocation (3919.86801405, 1444.88615894, 4805.73488539) km>),
'057': ('Belgrade',
 <EarthLocation (4245.75101131, 1588.54468594, 4472.09453892) km>),
'058': ('Kaliningrad',
 <EarthLocation (3459.00904936, 1292.92644919, 5183.00168894) km>),
'059': ('Lomnicky Stit',
 <EarthLocation (3920.21698036, 1443.92063325, 4806.81916868) km>),
'060': ('Warsaw-Ostrowik',
 <EarthLocation (3655.89219348, 1434.20082599, 5009.06989295) km>),
'061': ('Uzhgorod',
 <EarthLocation (3907.41760256, 1602.4292479, 4763.85603785) km>),
'062': ('Turku',
 <EarthLocation (2918.98519762, 1192.95746843, 5525.50764584) km>),
'063': ('Turku-Tuorla',
 <EarthLocation (2917.77460928, 1205.30170373, 5523.53042337) km>),
```

(continues on next page)

(continued from previous page)

```
'064': ('Turku-Kevola',
 <EarthLocation (2910.90543707, 1220.64395439, 5523.78554885) km>),
'065': ('Traunstein',
 <EarthLocation (4183.73435489, 937.61378087, 4707.065106) km>),
'066': ('Athens',
 <EarthLocation (4609.16749036, 2025.02666202, 3903.0754246) km>),
'067': ('Lvov University Observatory',
 <EarthLocation (3765.05478887, 1678.64947375, 4851.08343946) km>),
'068': ('Lvov Polytechnic Institute',
 <EarthLocation (3765.21746573, 1677.5010837, 4851.33856494) km>),
'069': ('Baldone',
 <EarthLocation (3190.19339219, 1447.41788659, 5312.15896319) km>),
'070': ('Vilnius (before 1939)',
 <EarthLocation (3341.40237253, 1578.51055665, 5181.15202921) km>),
'071': ('NAO Rozhen, Smolyan',
 <EarthLocation (4333.20994879, 1996.52032812, 4221.36997345) km>),
'072': ('Scheuren Observatory',
 <EarthLocation (3980.4762811, 500.73380032, 4936.678038) km>),
'073': ('Bucharest',
 <EarthLocation (4098.25839931, 2007.42338918, 4441.0967931) km>),
'074': ('Boyden Observatory, Bloemfontein',
 <EarthLocation (4999.62784208, 2482.46766734, -3078.28026031) km>),
'075': ('Tartu',
 <EarthLocation (2994.15353439, 1507.31691089, 5408.08614367) km>),
'076': ('Johannesburg-Hartbeespoort',
 <EarthLocation (5081.34789792, 2687.80145592, -2756.94971825) km>),
'077': ('Yale-Columbia Station, Johannesburg',
 <EarthLocation (5056.84019248, 2692.07626792, -2798.47139012) km>),
'078': ('Johannesburg',
 <EarthLocation (5054.96801869, 2696.26773337, -2797.96113916) km>),
'079': ('Radcliffe Observatory, Pretoria',
 <EarthLocation (5064.3460728, 2718.75691957, -2758.60803387) km>),
'080': ('Istanbul',
 <EarthLocation (4216.76456077, 2334.18801716, 4163.52027086) km>),
'081': ('Leiden Station, Johannesburg',
 <EarthLocation (5081.34789792, 2687.80145592, -2756.94971825) km>),
'082': ('St. Polten',
 <EarthLocation (4108.42796312, 1159.1687311, 4723.83960631) km>),
'083': ('Golosseevo-Kiev',
 <EarthLocation (3512.46821796, 2069.46434619, 4888.90579187) km>),
'084': ('Pulkovo',
 <EarthLocation (2778.58783161, 1625.45864227, 5487.81285617) km>),
'085': ('Kiev',
 <EarthLocation (3506.10276325, 2065.44194273, 4895.15636613) km>),
'086': ('Odessa',
 <EarthLocation (3781.13956622, 2250.27439734, 4601.95340824) km>),
'087': ('Helwan',
 <EarthLocation (4728.35569796, 2879.53301032, 3156.85890815) km>),
'088': ('Kottomia',
 <EarthLocation (4700.72465706, 2917.41311867, 3164.06620296) km>),
'089': ('Nikolaev',
 <EarthLocation (3698.53557769, 2308.83131472, 4639.64819791) km>),
```

(continues on next page)

(continued from previous page)

```
'090': ('Mainz', <EarthLocation (4071.32522216, 590.314063, 4860.140394) km>),
'091': ('Observatoire de Nurol, Aurec sur Loire',
 <EarthLocation (4475.74354109, 329.39983037, 4517.55152132) km>),
'092': ('Torun-Piwnice',
 <EarthLocation (3638.66660739, 1221.33771037, 5077.06083337) km>),
'093': ('Skibotn',
 <EarthLocation (2114.94262176, 785.05721461, 5945.6993114) km>),
'094': ('Crimea-Simeis',
 <EarthLocation (3784.26921659, 2552.27197141, 4440.4589794) km>),
'095': ('Crimea-Nauchnij',
 <EarthLocation (3762.66366421, 2539.47775322, 4466.22665288) km>),
'096': ('Merate',
 <EarthLocation (4402.30762245, 731.03150078, 4542.19026455) km>),
'097': ('Wise Observatory, Mitzpeh Ramon',
 <EarthLocation (4514.85942197, 3133.52866747, 3227.84757296) km>),
'098': ('Asiago Observatory, Cima Ekar',
 <EarthLocation (4360.96869577, 892.7196545, 4554.56385033) km>),
'099': ('Lahti',
 <EarthLocation (2785.89832875, 1330.91224682, 5562.46257162) km>),
'100': ('Ahtari',
 <EarthLocation (2684.11671342, 1202.99452494, 5640.63301869) km>),
'101': ('Kharkov',
 <EarthLocation (3313.39626548, 2427.90023426, 4863.07433702) km>),
'102': ('Zvenigorod',
 <EarthLocation (2886.26508098, 2156.02422093, 5245.81358212) km>),
'103': ('Ljubljana',
 <EarthLocation (4293.32563478, 1112.54743989, 4568.9529274) km>),
'104': ('San Marcello Pistoiese',
 <EarthLocation (4509.8709013, 860.60980038, 4413.92592948) km>),
'105': ('Moscow',
 <EarthLocation (2851.35670964, 2193.5103761, 5249.33431374) km>),
'106': ('Crni Vrh',
 <EarthLocation (4309.82043143, 1080.24133019, 4561.57980103) km>),
'107': ('Cavezzo',
 <EarthLocation (4445.10592117, 864.28260474, 4476.55923482) km>),
'108': ('Montelupo',
 <EarthLocation (4530.43575523, 882.90894752, 4387.13775408) km>),
'109': ('Algiers-Kouba',
 <EarthLocation (5110.5335864, 274.13768976, 3793.77966897) km>),
'110': ('Rostov',
 <EarthLocation (2676.44293014, 2199.62871721, 5337.41638571) km>),
'111': ('Piazzano Observatory, Florence',
 <EarthLocation (4535.80024488, 879.37832422, 4382.4179327) km>),
'112': ('Pleiade Observatory, Verona',
 <EarthLocation (4398.61968727, 847.35115334, 4525.2882015) km>),
'113': ('Volkssternwarte Drebach, Schoenbrunn',
 <EarthLocation (3946.17291639, 912.25012548, 4911.22927137) km>),
'114': ('Engelhardt Observatory, Zelenchukskaya Station',
 <EarthLocation (3466.62072975, 3059.21715925, 4381.90768174) km>),
'115': ('Zelenchukskaya',
 <EarthLocation (3466.4731984, 3060.48304896, 4381.21884294) km>),
'116': ('Giesing',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4179.44730706, 857.59804937, 4725.81682878) km>),
'117': ('Sendling',
 <EarthLocation (4180.55284763, 853.46889785, 4725.68926604) km>),
'118': ('Astronomical and Geophysical Observatory, Modra',
 <EarthLocation (4053.68520433, 1260.56467252, 4744.95123978) km>),
'119': ('Abastuman',
 <EarthLocation (3496.1527862, 3239.74057387, 4226.28113894) km>),
'120': ('Visnjan',
 <EarthLocation (4367.4876044, 1066.78670227, 4509.27907763) km>),
'121': ('Kharkov University, Chuguevskaya Station',
 <EarthLocation (3308.01251533, 2486.79976083, 4837.14083198) km>),
'122': ('Pises Observatory',
 <EarthLocation (4584.75827457, 280.69694452, 4412.14005112) km>),
'123': ('Byurakan',
 <EarthLocation (3485.79830439, 3400.66238966, 4107.39266526) km>),
'124': ('Castres',
 <EarthLocation (4622.73530048, 182.03179852, 4376.16735844) km>),
'125': ('Tbilisi',
 <EarthLocation (3384.03127049, 3359.25687083, 4222.49252556) km>),
'126': ('Monte Viseggi',
 <EarthLocation (4518.65498337, 779.71944701, 4418.96465771) km>),
'127': ('Bornheim',
 <EarthLocation (4012.82216199, 491.26948522, 4916.77825056) km>),
'128': ('Saratov',
 <EarthLocation (2761.18637793, 2859.9524091, 4971.07533084) km>),
'129': ('Ordubad',
 <EarthLocation (3447.57080049, 3560.10289251, 4005.470036) km>),
'130': ('Lumezzane',
 <EarthLocation (4394.48373366, 793.83076808, 4539.90051337) km>),
'131': ("Observatoire de l'Ardeche",
 <EarthLocation (4527.70727965, 374.234309, 4462.1446452) km>),
'132': ('Bedoin',
 <EarthLocation (4567.87772369, 419.41545029, 4417.4976862) km>),
'133': ('Les Tardieux',
 <EarthLocation (4626.17600127, 412.11019979, 4356.84160333) km>),
'134': ('Groszschwabhausen',
 <EarthLocation (3947.84895458, 801.93972486, 4928.8520639) km>),
'135': ('Kasan',
 <EarthLocation (2352.32037896, 2717.60496833, 5251.37531758) km>),
'136': ('Engelhardt Observatory, Kasan',
 <EarthLocation (2363.79043623, 2701.61989479, 5254.50060471) km>),
'137': ('Givatayim Observatory',
 <EarthLocation (4441.62457931, 3088.69982414, 3367.0185223) km>),
'138': ('Village-Neuf',
 <EarthLocation (4270.86522805, 567.70817227, 4687.54800678) km>),
'139': ('Antibes',
 <EarthLocation (4590.22880516, 572.62190341, 4376.55004666) km>),
'140': ('Augerolles',
 <EarthLocation (4452.24046596, 282.40490849, 4543.84858017) km>),
'141': ('Hottviller',
 <EarthLocation (4152.42707229, 536.88906519, 4795.65742893) km>),
'142': ('Sinsen',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (3933.24356876, 496.0054702, 4979.73046275) km>),
'143': ('Gnosca',
 <EarthLocation (4365.25187315, 693.26716767, 4583.26546683) km>),
'144': ('Bray et Lu',
 <EarthLocation (4179.03774748, 121.54891288, 4800.69615716) km>),
'145': ("'s-Gravenwezel",
 <EarthLocation (3988.59664113, 318.09141421, 4950.32725118) km>),
'146': ('Frignano',
 <EarthLocation (4495.03443002, 846.6888484, 4431.97605719) km>),
'147': ('Osservatorio Astronomico di Suno',
 <EarthLocation (4417.51206104, 666.02847697, 4537.3556367) km>),
'148': ('Guitalens',
 <EarthLocation (4620.01472522, 164.36201886, 4379.67533379) km>),
'149': ('Beine-Nauroy',
 <EarthLocation (4160.16410995, 307.22621047, 4808.86017252) km>),
'150': ('Maisons Laffitte',
 <EarthLocation (4194.2223368, 157.9881472, 4786.47291165) km>),
'151': ('Eschenberg Observatory, Winterthur',
 <EarthLocation (4269.01029075, 656.60589527, 4678.10836402) km>),
'152': ('Moletai Astronomical Observatory',
 <EarthLocation (3281.72338187, 1569.75458341, 5221.65319916) km>),
'153': ('Stuttgart-Hoffeld',
 <EarthLocation (4160.75363805, 672.00972238, 4771.73941518) km>),
'154': ('Cortina',
 <EarthLocation (4298.26922986, 921.80649187, 4608.2039825) km>),
'155': ('Ole Romer Observatory, Aarhus',
 <EarthLocation (3506.80203339, 630.79003665, 5272.42316968) km>),
'156': ('Catania Astrophysical Observatory',
 <EarthLocation (4891.62175946, 1318.56034774, 3861.89817213) km>),
'157': ('Frasso Sabino',
 <EarthLocation (4612.64126266, 1048.9569659, 4264.67752368) km>),
'158': ('Promiod',
 <EarthLocation (4417.28648448, 589.65080305, 4549.71646621) km>),
'159': ('Monte Agliale',
 <EarthLocation (4519.21337452, 838.83499624, 4408.24938755) km>),
'160': ('Castelmartini',
 <EarthLocation (4526.90021309, 866.94615924, 4393.98149508) km>),
'161': ('Cerrina Tololo Observatory',
 <EarthLocation (4465.26110635, 640.31196924, 4494.48179979) km>),
'162': ('Potenza',
 <EarthLocation (4663.95261747, 1318.05129925, 4133.54302696) km>),
'163': ('Roeser Observatory, Luxembourg',
 <EarthLocation (4123.01358587, 444.20424703, 4830.22693147) km>),
'164': ('St. Michel sur Meurthe',
 <EarthLocation (4219.16023493, 509.53595917, 4740.55032525) km>),
'165': ('Piera Observatory, Barcelona',
 <EarthLocation (4780.33807719, 146.49514477, 4206.12622602) km>),
'166': ('Upice',
 <EarthLocation (3907.09489405, 1121.20492664, 4899.17459244) km>),
'167': ('Bulach Observatory',
 <EarthLocation (4267.35937507, 643.29778183, 4681.42499526) km>),
'168': ('Kourovskaya',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (1763.10375033, 2998.80255711, 5328.35943117) km>),
'169': ('Airali Observatory',
 <EarthLocation (4463.28431779, 659.20784324, 4493.58886061) km>),
'170': ('Observatorio de Begues',
 <EarthLocation (4794.74825142, 160.78399238, 4191.13760407) km>),
'171': ('Flarestar Observatory, San Gwann',
 <EarthLocation (5007.91226099, 1292.30906934, 3720.17596799) km>),
'172': ('Onnens',
 <EarthLocation (4342.00565647, 535.93153922, 4626.63679843) km>),
'173': ('St. Clotilde, Reunion',
 <EarthLocation (3375.97289308, 4913.19155424, -2260.85822239) km>),
'174': ('Nyrola Observatory, Jyvaskyla',
 <EarthLocation (2678.69808374, 1278.42531638, 5626.72868003) km>),
'175': ('F.-X. Bagnoud Observatory, St-Luc',
 <EarthLocation (4382.40086669, 585.3832785, 4584.6048756) km>),
'176': ('Observatorio Astronomico de Consell',
 <EarthLocation (4911.45063407, 242.1433835, 4048.52246075) km>),
'177': ('Le Cres',
 <EarthLocation (4611.74912188, 317.74546113, 4379.80289653) km>),
'178': ('Collonges',
 <EarthLocation (4402.53977183, 473.16938949, 4575.99439065) km>),
'179': ('Monte Generoso',
 <EarthLocation (4390.23867712, 696.72012228, 4560.81442459) km>),
'180': ('Mauguio',
 <EarthLocation (4617.67200771, 319.00380324, 4373.4885409) km>),
'181': ('Observatoire des Makes, Saint-Louis',
 <EarthLocation (3377.83611802, 4898.28100984, -2292.36621917) km>),
'182': ('St. Paul, Reunion',
 <EarthLocation (3394.62048214, 4894.89381861, -2272.78533858) km>),
'183': ('Starlab Observatory, Karachay-Cherkessia',
 <EarthLocation (3467.36662641, 3059.04662377, 4381.46121215) km>),
'184': ('Valmeca Observatory, Puimichel',
 <EarthLocation (4571.93606404, 483.44247047, 4407.10132289) km>),
'185': ('Observatoire Astronomique Jurassien-Vicques',
 <EarthLocation (4292.95342645, 559.22594656, 4668.796284) km>),
'186': ('Kitab',
 <EarthLocation (1945.24731442, 4556.62331031, 4004.25818997) km>),
'187': ('Astronomical Observatory, Borowiec',
 <EarthLocation (3738.34651581, 1148.15887443, 5021.82616695) km>),
'188': ('Majdanak',
 <EarthLocation (1957.36504855, 4587.99069389, 3965.68321739) km>),
'189': ('Geneva (before 1967)',
 <EarthLocation (4397.13581302, 473.90836004, 4580.96933751) km>),
'190': ('Gissar',
 <EarthLocation (1817.47733136, 4657.24060552, 3948.64083533) km>),
'191': ('Dushanbe',
 <EarthLocation (1807.65679652, 4655.86159007, 3954.82762822) km>),
'192': ('Tashkent',
 <EarthLocation (1696.18659124, 4487.31161158, 4189.92575804) km>),
'193': ('Sanglok',
 <EarthLocation (1779.85803661, 4689.89788737, 3929.5702057) km>),
'194': ('Tivoli',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (5568.68699679, 1810.38619709, -2524.40284323) km>),
'195': ('Untermenzing Observatory, Munich',
 <EarthLocation (4176.06433452, 845.77471169, 4730.91933838) km>),
'196': ('Homburg-Erbach',
 <EarthLocation (4130.6049774, 531.56830752, 4814.8556213) km>),
'197': ('Bastia',
 <EarthLocation (4472.55219199, 965.66005476, 4428.59564458) km>),
'198': ('Wildberg',
 <EarthLocation (4174.33884052, 642.99488933, 4763.98360059) km>),
'199': ('Buthiers',
 <EarthLocation (4247.75395129, 180.85588418, 4738.57310278) km>),
'200': ('Beersel Hills Observatory',
 <EarthLocation (4031.38319183, 303.37595738, 4916.77825056) km>),
'201': ('Jonathan B. Postel Observatory',
 <EarthLocation (4417.28648448, 589.65080305, 4549.65268484) km>),
'202': ('Tamaris Observatoire, La Seyne sur Mer',
 <EarthLocation (4640.07038583, 479.48007475, 4335.28350027) km>),
'203': ('GiaGa Observatory',
 <EarthLocation (4419.92542661, 699.69157103, 4529.81667877) km>),
'204': ('Schiaparelli Observatory',
 <EarthLocation (4397.67338105, 678.50106407, 4555.9032591) km>),
'205': ('Obs. Casalecchio di Reno, Bologna',
 <EarthLocation (4471.00658753, 891.21255687, 4445.75283311) km>),
'206': ('Haagaar Observatory, Eina',
 <EarthLocation (3086.08275647, 575.6884584, 5534.3094749) km>),
'207': ('Osservatorio Antonio Grosso',
 <EarthLocation (4415.74771667, 723.62081369, 4530.07180425) km>),
'208': ('Rivalta',
 <EarthLocation (4458.49601366, 753.09767275, 4483.44762278) km>),
'209': ('Asiago Observatory, Cima Ekar-ADAS',
 <EarthLocation (4360.89636176, 892.691366, 4554.6276317) km>),
'210': ('Alma-Ata',
 <EarthLocation (1051.36637156, 4538.53388958, 4343.76642248) km>),
'211': ('Scandicci',
 <EarthLocation (4526.31606294, 894.29739846, 4389.11497655) km>),
'212': ('Observatorio La Dehesilla',
 <EarthLocation (5106.44864578, -414.67032712, 3786.750962) km>),
'213': ('Observatorio Montcabre',
 <EarthLocation (4778.45714169, 199.05609467, 4205.87747868) km>),
'214': ('Garching Observatory',
 <EarthLocation (4167.03666883, 859.68328736, 4736.27697346) km>),
'215': ('Buchloe',
 <EarthLocation (4199.91123573, 796.07189177, 4718.60953397) km>),
'216': ('Observatoire des Cote de Meuse',
 <EarthLocation (4171.81005687, 415.76959074, 4790.87382618) km>),
'217': ('Assah',
 <EarthLocation (978.43821791, 4552.81656137, 4347.61243909) km>),
'218': ('Hyderabad',
 <EarthLocation (1218.44056565, 5964.36554618, 1898.64382216) km>),
'219': ('Japal-Rangapur',
 <EarthLocation (1192.052327, 5981.01219924, 1863.43650592) km>),
'220': ('Vainu Bappu Observatory, Kavalur',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (1206.6513923, 6108.75020126, 1379.84615858) km>),
'221': ('IAS Observatory, Hakos',
 <EarthLocation (5627.99307234, 1652.32126829, -2501.52446581) km>),
'222': ('Yerres-Canotiers',
 <EarthLocation (4212.78383538, 183.48474723, 4769.37950449) km>),
'223': ('Madras',
 <EarthLocation (1052.72727727, 6124.20635555, 1432.84847705) km>),
'224': ('Ottmarsheim',
 <EarthLocation (4256.87225416, 560.55711361, 4700.99311958) km>),
'225': ('Northwood Ridge Observatory',
 <EarthLocation (1501.99340451, -4405.77430996, 4346.0625518) km>),
'226': ('Guido Ruggieri Observatory, Padua',
 <EarthLocation (4387.26036097, 923.40521684, 4521.33375656) km>),
'227': ('OrbitJet Observatory, Colden',
 <EarthLocation (919.6866232, -4608.73539256, 4298.35408704) km>),
'228': ('Bruno Zugna Observatory, Trieste',
 <EarthLocation (4336.77444055, 1071.23522232, 4537.85313139) km>),
'229': ('G. C. Gloriosi Astronomical Observatory, Salerno',
 <EarthLocation (4678.83239853, 1251.44026533, 4136.66831409) km>),
'230': ('Mt. Wendelstein Observatory',
 <EarthLocation (4207.21163234, 895.29122787, 4696.2222731) km>),
'231': ('Vesqueville',
 <EarthLocation (4089.49287071, 386.44847012, 4863.52080661) km>),
'232': ('Masquefa Observatory',
 <EarthLocation (4782.31071852, 111.1731142, 4205.1057241) km>),
'233': ('Sauro Donati Astronomical Observatory, San Vito',
 <EarthLocation (4528.94246334, 842.68523675, 4396.51361547) km>),
'234': ('Coddenham Observatory',
 <EarthLocation (3921.48119457, 77.23600233, 5012.77559055) km>),
'235': ('CAST Observatory, Talmassons',
 <EarthLocation (4327.57606134, 1008.13502559, 4560.32330804) km>),
'236': ('Tomsk',
 <EarthLocation (311.0819235, 3517.84681041, 5293.53480315) km>),
'237': ('Baugy',
 <EarthLocation (4346.21486379, 207.49397418, 4648.3862456) km>),
'238': ('Grorudalen Optical Observatory',
 <EarthLocation (3144.21087503, 606.01456746, 5497.76274989) km>),
'239': ('Trebur',
 <EarthLocation (4070.02475063, 601.83645131, 4857.52535783) km>),
'240': ('Herrenberg Sternwarte',
 <EarthLocation (4174.18958183, 648.6733701, 4763.39681198) km>),
'241': ('Schaerding',
 <EarthLocation (4122.61613213, 987.46983836, 4750.05374938) km>),
'242': ('Varennes',
 <EarthLocation (4633.66394667, 137.16778294, 4366.4725902) km>),
'243': ('Umbrella Observatory, Fredenbeck',
 <EarthLocation (3748.42267726, 621.42118206, 5105.6986685) km>),
'244': ('Geocentric Occultation Observation',
 <EarthLocation (0., 0., 0.) km>),
'245': ('Spitzer Space Telescope', <EarthLocation (0., 0., 0.) km>),
'246': ('Klet Observatory-KLENOT',
 <EarthLocation (4074.57052676, 1037.44087831, 4781.45969597) km>),
```

(continues on next page)

(continued from previous page)

```
'247': ('Roving Observer', <EarthLocation (0., 0., 0.) km>),
'248': ('Hipparcos', <EarthLocation (0., 0., 0.) km>),
'249': ('SOHO', <EarthLocation (0., 0., 0.) km>),
'250': ('Hubble Space Telescope', <EarthLocation (0., 0., 0.) km>),
'251': ('Arecibo',
 <EarthLocation (2390.47973654, -5564.81705856, 1994.66029656) km>),
'252': ('Goldstone DSS 13, Fort Irwin',
 <EarthLocation (-2351.14653742, -4655.51270722, 3660.91669712) km>),
'253': ('Goldstone DSS 14, Fort Irwin',
 <EarthLocation (-2353.62433941, -4641.34887792, 3677.05976187) km>),
'254': ('Haystack, Westford',
 <EarthLocation (1492.37277526, -4457.31561616, 4296.8998718) km>),
'255': ('Evpatoria',
 <EarthLocation (3768.29743637, 2464.67347364, 4502.23761438) km>),
'256': ('Green Bank',
 <EarthLocation (882.59106092, -4924.87600104, 3943.72966984) km>),
'257': ('Goldstone DSS 25, Fort Irwin',
 <EarthLocation (-2355.02473855, -4646.953190 5, 3669.03606552) km>),
'258': ('Gaia', <EarthLocation (0., 0., 0.) km>),
'259': ('EISCAT Tromso UHF',
 <EarthLocation (2106.80923363, 734.73497155, 5955.18997926) km>),
'260': ('Siding Spring Observatory-DSS',
 <EarthLocation (-4680.91624241, 2805.23346857, -3292.77700762) km>),
'261': ('Palomar Mountain-DSS',
 <EarthLocation (-2410.03526915, -4758.71525202, 3488.05642815) km>),
'262': ('European Southern Observatory, La Silla-DSS',
 <EarthLocation (1838.1727099, -5258.92294357, -3100.10624512) km>),
'266': ('New Horizons KBO Search-Subaru',
 <EarthLocation (-5464.47910846, -2493.00694552, 2150.94378747) km>),
'267': ('New Horizons KBO Search-CFHT',
 <EarthLocation (-5464.19769922, -2493.68513119, 2150.92465306) km>),
'268': ('New Horizons KBO Search-Magellan/Clay',
 <EarthLocation (1846.38419885, -5270.07774077, -3076.44335685) km>),
'269': ('New Horizons KBO Search-Magellan/Baade',
 <EarthLocation (1846.47364237, -5270.00585336, -3076.48800381) km>),
'270': ('Unistellar Network, Roving Observer',
 <EarthLocation (0., 0., 0.) km>),
'275': ('Non-geocentric Occultation Observation',
 <EarthLocation (0., 0., 0.) km>),
'276': ('Plonsk',
 <EarthLocation (3636.86226246, 1351.29359379, 5045.59110541) km>),
'277': ('Royal Observatory, Blackford Hill, Edinburgh',
 <EarthLocation (3576.31015473, -198.85106188, 5259.85823979) km>),
'278': ('Peking, Transit of Venus site',
 <EarthLocation (-2182.29853019, 4386.71114522, 4069.82543833) km>),
'279': ('Seeberg Observatory, Gotha (1787-1857)',
 <EarthLocation (3957.5555746, 749.80689217, 4929.1964833) km>),
'280': ('Lilienthal',
 <EarthLocation (3787.86731031, 593.963454, 5079.93099502) km>),
'281': ('Bologna',
 <EarthLocation (4467.89595283, 897.00753736, 4447.66627421) km>),
'282': ('Nimes',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4594.548341, 350.32900623, 4395.30176944) km>),
'283': ('Bremen',
 <EarthLocation (3794.52456051, 588.5288566, 5075.65764323) km>),
'284': ('Driesen',
 <EarthLocation (3714.61889355, 1053.3092656, 5059.71230073) km>),
'285': ('Flammarion Observatory, Juvisy',
 <EarthLocation (4214.57031797, 174.49120834, 4768.23143983) km>),
'286': ('Yunnan Observatory',
 <EarthLocation (-1280.41478019, 5641.09841739, 2682.45307809) km>),
'290': ('Mt. Graham-VATT',
 <EarthLocation (-1828.87960609, -5054.42630114, 3427.85319301) km>),
'291': ('LPL/Spacewatch II',
 <EarthLocation (-1994.49333931, -5037.57679206, 3357.89778639) km>),
'292': ('Burlington, New Jersey',
 <EarthLocation (1273.7107536, -4718.68266208, 4084.11246521) km>),
'293': ('Burlington remote site',
 <EarthLocation (1318.77884203, -4726.55166779, 4060.83226516) km>),
'294': ('Astrophysical Obs., College of Staten Island',
 <EarthLocation (1324.18809249, -4665.06498621, 4129.14211243) km>),
'295': ('Catholic University Observatory, Washington',
 <EarthLocation (1117.54129595, -4840.60316194, 3986.9734387) km>),
'296': ('Dudley Observatory, Albany (after 1893)',
 <EarthLocation (1314.61427259, -4509.79776038, 4300.1399654) km>),
'297': ('Middlebury',
 <EarthLocation (1329.32020108, -4397.64993079, 4409.2061081) km>),
'298': ('Van Vleck Observatory',
 <EarthLocation (1424.69159002, -4562.71195894, 4208.80504356) km>),
'299': ('Bosscha Observatory, Lembang',
 <EarthLocation (-1917.05131278, 6037.46122824, -753.13041696) km>),
'300': ('Bisei Spaceguard Center-BATTeRS',
 <EarthLocation (-3617.89347943, 3806.54688316, 3608.23966364) km>),
'301': ('Mont Megantic',
 <EarthLocation (1448.01118609, -4242.16787663, 4523.75744862) km>),
'302': ('University of the Andes station',
 <EarthLocation (2043.24435743, -5974.63300278, 956.72055) km>),
'303': ('OAN de Llano del Hato, Merida',
 <EarthLocation (2066.95326128, -5959.04673968, 968.52010345) km>),
'304': ('Las Campanas Observatory',
 <EarthLocation (1845.61759041, -5270.84634425, -3075.34631729) km>),
'305': ('Purple Mountain, Hainan Island station',
 <EarthLocation (-1751.66483484, 4932.48187266, 3633.17817931) km>),
'306': ('Observatorio Taya Beixo, Barquisimeto',
 <EarthLocation (2217.80575664, -5876.41669462, 1108.58399197) km>),
'307': ('Shattuck Observatory, Hanover',
 <EarthLocation (1405.42369129, -4399.37335935, 4384.52271791) km>),
'309': ('Cerro Paranal',
 <EarthLocation (1946.46278, -5467.60400814, -2642.69177203) km>),
'310': ('Minor Planet Center Test Code',
 <EarthLocation (1526.21345383, -4464.91510502, 4277.01284064) km>),
'312': ('Tsingtao field station, Xisha Islands',
 <EarthLocation (-2320.47406261, 5648.35086437, 1834.9900149) km>),
'318': ('Quinns Rock',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-2355.97508455, 4897.32313917, -3327.4740729) km>),
'319': ('Perth Observatory, Perth-Lowell Telescope',
 <EarthLocation (-2384.77995038, 4860.42414035, -3361.40576174) km>),
'320': ('Chiro Observatory',
 <EarthLocation (-2438.17444295, 4903.48067394, -3259.35556974) km>),
'321': ('Craigie',
 <EarthLocation (-2358.07564525, 4887.24900564, -3340.74059786) km>),
'322': ('Perth Observatory, Bickley-MCT',
 <EarthLocation (-2384.66702571, 4860.4085011, -3361.46954311) km>),
'323': ('Perth Observatory, Bickley',
 <EarthLocation (-2384.75185548, 4860.36688007, -3361.46954311) km>),
'324': ('Peking Observatory, Shaho Station',
 <EarthLocation (-2166.7527395, 4378.76018951, 4086.59993864) km>),
'327': ('Peking Observatory, Xinglong Station',
 <EarthLocation (-2252.10719413, 4312.46570607, 4111.9849239) km>),
'330': ('Purple Mountain Observatory, Nanking',
 <EarthLocation (-2608.23156103, 4740.25892027, 3366.89095956) km>),
'333': ('Desert Eagle Observatory',
 <EarthLocation (-1895.10026885, -5075.04752992, 3357.57887954) km>),
'334': ('Tsingtao',
 <EarthLocation (-2605.64736811, 4455.53140436, 3734.52677624) km>),
'337': ('Sheshan, formerly Zo-Se',
 <EarthLocation (-2830.55142042, 4676.68752817, 3275.04578676) km>),
'340': ('Toyonaka',
 <EarthLocation (-3738.46145447, 3675.66171508, 3620.35812394) km>),
'341': ('Akashina',
 <EarthLocation (-3820.52333138, 3446.22574412, 3758.18966451) km>),
'342': ('Shishikui',
 <EarthLocation (-3717.49633974, 3806.94690226, 3505.10518835) km>),
'343': ('Younchun',
 <EarthLocation (-3029.19960095, 4001.57327978, 3923.00072459) km>),
'344': ('Bohyunsan Optical Astronomy Observatory',
 <EarthLocation (-3243.24037109, 4008.40017412, 3743.64751215) km>),
'345': ('Sobaeksan Optical Astronomy Observatory',
 <EarthLocation (-3175.24869038, 3997.91814889, 3812.40382901) km>),
'346': ('KNUE Astronomical Observatory',
 <EarthLocation (-3116.46433541, 4078.31952774, 3773.68853742) km>),
'347': ('Utsunomiya-Imaizumi',
 <EarthLocation (-3923.8591243, 3303.18972452, 3778.66348428) km>),
'348': ('Ayabe',
 <EarthLocation (-3701.72321562, 3667.39552108, 3665.83424075) km>),
'349': ('Ageo',
 <EarthLocation (-3934.30530097, 3352.35791856, 3724.29624449) km>),
'350': ('Kurohone',
 <EarthLocation (-3890.62476273, 3350.77978088, 3771.51997084) km>),
'351': ('Sakamoto',
 <EarthLocation (-3751.01394535, 3639.07533542, 3644.14857495) km>),
'352': ('Konan',
 <EarthLocation (-3776.26240392, 3624.11519953, 3633.17817931) km>),
'353': ('Nishi Kobe',
 <EarthLocation (-3714.36491212, 3705.97270441, 3614.42645653) km>),
'354': ('Kawachi',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-3915.25537234, 3282.8912473, 3806.08947338) km>),
'355': ('Hadano',
 <EarthLocation (-3941.48459246, 3400.60193147, 3673.1690983) km>),
'356': ('Kogota',
 <EarthLocation (-3886.82438307, 3137.761868, 3952.5314989) km>),
'357': ('Shimotsuma',
 <EarthLocation (-3948.54866339, 3312.47419735, 3744.73179544) km>),
'358': ('Nanyo',
 <EarthLocation (-3861.78828402, 3222.25095354, 3909.54285552) km>),
'359': ('Wakayama',
 <EarthLocation (-3744.67247389, 3722.26992939, 3566.14395944) km>),
'360': ('Kuma Kogen',
 <EarthLocation (-3620.27158149, 3889.85418724, 3516.77717906) km>),
'361': ('Sumoto',
 <EarthLocation (-3720.54163501, 3734.42478388, 3578.51754522) km>),
'362': ('Ray Observatory',
 <EarthLocation (-3633.9093117, 2979.08945531, 4298.73677526) km>),
'363': ('Yamada',
 <EarthLocation (-3474.36215368, 4029.30799272, 3505.87056479) km>),
'364': ('YCPM Kagoshima Station',
 <EarthLocation (-3535.13655686, 4128.20239082, 3327.09138468) km>),
'365': ('Uto Observatory',
 <EarthLocation (-3786.89982745, 3662.34790819, 3584.25786852) km>),
'366': ('Miyasaka Observatory',
 <EarthLocation (-3864.36850261, 3442.99044808, 3716.34908579) km>),
'367': ('Yatsuka',
 <EarthLocation (-3556.39166954, 3791.54628793, 3683.18277339) km>),
'368': ('Ochiai',
 <EarthLocation (-3898.11287782, 3411.13405334, 3711.50170167) km>),
'369': ('Chichibu',
 <EarthLocation (-3908.39178234, 3379.58972829, 3727.3832628) km>),
'370': ('Kochi',
 <EarthLocation (-3664.50241104, 3857.89671711, 3505.16896972) km>),
'371': ('Tokyo-Okayama',
 <EarthLocation (-3625.57296162, 3807.69239826, 3599.24649047) km>),
'372': ('Geisei',
 <EarthLocation (-3685.82045399, 3839.83369231, 3502.8728404) km>),
'373': ('Oishi',
 <EarthLocation (-3759.36848595, 3715.05299275, 3558.80910189) km>),
'374': ('Minami-Oda Observatory',
 <EarthLocation (-3676.2619608, 3712.42175251, 3646.63604838) km>),
'375': ('Uzurano',
 <EarthLocation (-3692.57077056, 3709.26167486, 3633.6246489) km>),
'376': ('Uenohara',
 <EarthLocation (-3916.82826769, 3400.14489483, 3700.72265014) km>),
'377': ('Kwasan Observatory, Kyoto',
 <EarthLocation (-3749.70801975, 3647.28495166, 3637.51531247) km>),
'378': ('Murou',
 <EarthLocation (-3783.15406315, 3651.53787335, 3598.92758362) km>),
'379': ('Hamamatsu-Yuto',
 <EarthLocation (-3878.02787374, 3537.66467111, 3610.85469981) km>),
'380': ('Ishiki',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-3836.07263952, 3572.82956368, 3620.99593764) km>),
'381': ('Tokyo-Kiso',
 <EarthLocation (-3826.85454143, 3491.2861695, 3710.65340945) km>),
'382': ('Tokyo-Norikura',
 <EarthLocation (-3808.35557059, 3482.95885659, 3740.07575543) km>),
'383': ('Chirorin',
 <EarthLocation (-3796.01006666, 3430.45174366, 3796.64983062) km>),
'384': ('Shimada',
 <EarthLocation (-3906.65871556, 3495.50884234, 3621.5061886) km>),
'385': ('Nihondaira Observatory',
 <EarthLocation (-3917.01854628, 3469.38725271, 3635.34674589) km>),
'386': ('Yatsugatake-Kobuchizawa',
 <EarthLocation (-3864.41561207, 3440.44417167, 3719.02790333) km>),
'387': ('Tokyo-Dodaira',
 <EarthLocation (-3910.52676006, 3376.14319091, 3729.23292253) km>),
'388': ('Tokyo-Mitaka',
 <EarthLocation (-3946.95773858, 3366.00782309, 3698.74542767) km>),
'389': ('Tokyo (before 1938)',
 <EarthLocation (-3959.66286212, 3352.73087779, 3697.08711205) km>),
'390': ('Utsunomiya',
 <EarthLocation (-3922.1672692, 3305.99020253, 3778.02567058) km>),
'391': ('Sendai Observatory, Ayashi Station',
 <EarthLocation (-3884.64534073, 3170.67539605, 3928.67726652) km>),
'392': ('JCPM Sapporo Station',
 <EarthLocation (-3654.78899015, 2921.05912357, 4320.61378517) km>),
'393': ('JCPM Sakura Station',
 <EarthLocation (-3960.19160377, 3307.80635854, 3736.3126546) km>),
'394': ('JCPM Hamatonbetsu Station',
 <EarthLocation (-3568.49519925, 2755.97777572, 4496.14011541) km>),
'395': ('Tokyo-Asahikawa',
 <EarthLocation (-3648.47993085, 2813.94033236, 4395.1742067) km>),
'396': ('Asahikawa',
 <EarthLocation (-3657.61298547, 2814.62649038, 4387.5204423) km>),
'397': ('Sapporo Science Center',
 <EarthLocation (-3653.12437436, 2908.31516487, 4330.24477204) km>),
'398': ('Nagatoro',
 <EarthLocation (-3899.1632358, 3376.60832603, 3739.5017231) km>),
'399': ('Kushiro',
 <EarthLocation (-3803.01022567, 2703.65778793, 4333.9440915) km>),
'400': ('Kitami',
 <EarthLocation (-3722.65279976, 2726.29648499, 4388.85985107) km>),
'401': ('Oosato',
 <EarthLocation (-3918.0236829, 3355.68595659, 3738.2260957) km>),
'402': ('Dynic Astronomical Observatory',
 <EarthLocation (-3772.43877872, 3604.0383729, 3656.90484895) km>),
'403': ('Kani',
 <EarthLocation (-3809.48990081, 3545.50168074, 3675.40144625) km>),
'404': ('Yamamoto',
 <EarthLocation (-3916.36255669, 3179.42877099, 3890.0257563) km>),
'405': ('Kamihoriguchi',
 <EarthLocation (-3903.4624235, 3354.04774922, 3754.8092519) km>),
'406': ('Bibai',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-3657.44423966, 2875.71720811, 4348.23111838) km>),
'407': ('Kahoku',
 <EarthLocation (-3849.1791829, 3194.52677855, 3944.04857669) km>),
'408': ('Nyukasa',
 <EarthLocation (-3855.57598251, 3450.34751357, 3720.23974936) km>),
'409': ('Kiyose and Mizuho',
 <EarthLocation (-3941.06632859, 3363.47936956, 3707.22834988) km>),
'410': ('Sengamine',
 <EarthLocation (-3685.91072392, 3699.96169017, 3649.69755414) km>),
'411': ('Oizumi',
 <EarthLocation (-3910.97112223, 3350.09530339, 3750.66346285) km>),
'412': ('Iwaki',
 <EarthLocation (-3943.37174192, 3239.22580356, 3814.31727011) km>),
'413': ('Siding Spring Observatory',
 <EarthLocation (-4680.88790863, 2805.21870914, -3292.78976389) km>),
'414': ('Mount Stromlo',
 <EarthLocation (-4466.67818513, 2683.0340862, -3667.36499363) km>),
'415': ('Kambah',
 <EarthLocation (-4464.9718206, 2676.08460323, -3674.18960022) km>),
'416': ('Barton',
 <EarthLocation (-4472.94624848, 2673.44172999, -3666.47205445) km>),
'417': ('Yanagida Astronomical Observatory',
 <EarthLocation (-3721.86944624, 3454.08618532, 3847.10089429) km>),
'418': ('Tamworth',
 <EarthLocation (-4779.40882093, 2655.77906321, -3273.85945328) km>),
'419': ('Windsor',
 <EarthLocation (-4643.21080079, 2591.50495261, -3510.39904206) km>),
'420': ('Sydney',
 <EarthLocation (-4646.30464511, 2553.79960868, -3533.74302348) km>),
'421': ('Mt. Kajigamori, Otoyo',
 <EarthLocation (-3672.53469597, 3834.3697052, 3524.68606894) km>),
'422': ('Loomberah',
 <EarthLocation (-4771.86297885, 2640.06995603, -3298.07086133) km>),
'423': ('North Ryde',
 <EarthLocation (-4645.77735908, 2561.99123755, -3528.52570741) km>),
'424': ('Macquarie',
 <EarthLocation (-4472.89795777, 2680.60170342, -3661.36954485) km>),
'425': ('Taylor Range Observatory, Brisbane',
 <EarthLocation (-5043.1695586, 2577.21145298, -2923.92934491) km>),
'426': ('Woomera',
 <EarthLocation (-3982.19168817, 3736.68972347, -3284.61299226) km>),
'427': ('Stockport',
 <EarthLocation (-3962.84452839, 3477.97743737, -3577.11435508) km>),
'428': ('Reedy Creek',
 <EarthLocation (-5033.99686755, 2521.16753598, -2987.32802669) km>),
'429': ('Hawker',
 <EarthLocation (-4471.85452561, 2682.71398991, -3661.17820074) km>),
'430': ('Rainbow Observatory, near Coonabarabran',
 <EarthLocation (-4687.5111033, 2792.95371878, -3292.58566351) km>),
'431': ('Mt. Tarana Observatory, Bathurst',
 <EarthLocation (-4603.57715287, 2683.88705529, -3494.77260641) km>),
'432': ('Boambee',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-4912.46966257, 2494.15520383, -3202.88354474) km>),
'433': ('Bagnall Beach Observatory',
 <EarthLocation (-4746.34024364, 2512.23057019, -3429.58804627) km>),
'434': ('S. Benedetto Po',
 <EarthLocation (4431.75235776, 855.07357317, 4491.42029403) km>),
'435': ('G. Colombo Astronomical Observatory, Padua',
 <EarthLocation (4389.44386007, 924.48883648, 4519.03762724) km>),
'436': ('Osservatorio di Livergnano',
 <EarthLocation (4481.288456, 898.34572036, 4434.59109336) km>),
'437': ('Haverford',
 <EarthLocation (1241.15224668, -4731.96673472, 4079.00995561) km>),
'438': ('Smith College Observatory, Northampton',
 <EarthLocation (1409.56255961, -4508.36819017, 4271.94859986) km>),
'439': ('ROTSE-III, Los Alamos',
 <EarthLocation (-1491.43591001, -4956.72153803, 3717.68849456) km>),
'440': ('Elginfield Observatory',
 <EarthLocation (703.24797452, -4604.23737812, 4343.31995289) km>),
'441': ('Swilken Brae, St. Andrews',
 <EarthLocation (3539.30649022, -174.97720022, 5285.37078779) km>),
'442': ('Gualba Observatory',
 <EarthLocation (4764.3292128, -209.49807497, 4221.6888803) km>),
'443': ('Obs. Astronomico Plomer, Buenos Aires',
 <EarthLocation (2742.34583951, -4481.13857948, -3604.47656281) km>),
'444': ('Star Cruiser Observatory',
 <EarthLocation (-2394.8693678, -4757.40578858, 3499.55620916) km>),
'445': ("Observatorio d'Ontinyent",
 <EarthLocation (4975.96752461, -50.37299037, 3976.7684195) km>),
'446': ('Kingsnake Observatory, Seguin',
 <EarthLocation (-756.71350772, -5500.29533682, 3128.98644946) km>),
'447': ('Centennial Observatory',
 <EarthLocation (-1256.580445, -4757.8489593, 4046.80036376) km>),
'448': ('Desert Moon Observatory, Las Cruces',
 <EarthLocation (-1551.31880023, -5164.29744384, 3397.50601716) km>),
'449': ('Griffin Hunter Observatory, Bethune',
 <EarthLocation (883.33638725, -5194.8591271, 3581.70661372) km>),
'450': ('Carla Jane Observatory, Charlotte',
 <EarthLocation (846.77404041, -5151.82582943, 3651.73855798) km>),
'451': ('West Skies Observatory, Mulvane',
 <EarthLocation (-638.87491363, -5026.80265028, 3861.06901432) km>),
'452': ('Big Cypress Observatory, Fort Lauderdale',
 <EarthLocation (906.48277779, -5655.37912176, 2796.8130745) km>),
'453': ('Edwards Raven Observatory',
 <EarthLocation (-2445.53051028, -4625.26277491, 3636.87749877) km>),
'454': ('Maryland Space Grant Consortium Observatory',
 <EarthLocation (1142.91827099, -4806.10773163, 4020.93701822) km>),
'455': ('CBA Concord',
 <EarthLocation (-2669.85597353, -4266.62869363, 3904.56790866) km>),
'456': ('Daventry Observatory',
 <EarthLocation (3912.04063125, -80.04662771, 5020.16785133) km>),
'457': ('Partizanske',
 <EarthLocation (4009.12240943, 1329.01950848, 4763.51161845) km>),
'458': ('Guadarrama Observatory',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4834.95237187, -339.73793449, 4133.35168285) km>),
'459': ('Smith River Observatory, Danbury',
 <EarthLocation (1440.05579154, -4401.38147317, 4371.44753706) km>),
'460': ('Area 52 Observatory, Nashville',
 <EarthLocation (-369.50020306, -5281.58215826, 3544.90476323) km>),
'461': ('University of Szeged, Piszkesteto Stn. (Konkoly)',
 <EarthLocation (4027.50281479, 1457.48250697, 4711.46602053) km>),
'462': ('Mount Belleview Observatory',
 <EarthLocation (1124.87026343, -4839.8875161, 3985.57024856) km>),
'463': ('Sommers-Bausch Observatory, Boulder',
 <EarthLocation (-1288.22307704, -4721.08856036, 4079.39264383) km>),
'464': ('Toby Point Observatory, Narragansett',
 <EarthLocation (1520.1684596, -4542.96206121, 4196.75036463) km>),
'465': ('Takapuna',
 <EarthLocation (-5091.89257702, 465.18174858, -3799.96646186) km>),
'466': ('Mount Molehill Observatory, Auckland',
 <EarthLocation (-5083.1714182, 458.2488196, -3812.2124849) km>),
'467': ('Auckland Observatory',
 <EarthLocation (-5085.00436863, 464.86567972, -3809.27854188) km>),
'468': ('Astronomical Observatory, Campo Catino',
 <EarthLocation (4633.13403418, 1097.75409495, 4231.83011813) km>),
'469': ('Courroux',
 <EarthLocation (4293.15206464, 556.21167228, 4669.11519085) km>),
'470': ('Ceccano',
 <EarthLocation (4650.22819834, 1101.62944647, 4210.13169606) km>),
'471': ('Houstrup',
 <EarthLocation (3557.87002234, 515.16285967, 5250.80128525) km>),
'472': ('Merlette',
 <EarthLocation (4515.2168328, 500.10389055, 4464.56833726) km>),
'473': ('Remanzacco',
 <EarthLocation (4312.33179685, 1020.71213031, 4571.98892061) km>),
'474': ('Mount John Observatory, Lake Tekapo',
 <EarthLocation (-4533.67638431, 761.52780086, -4407.79653982) km>),
'475': ('Turin (before 1913)',
 <EarthLocation (4471.69063357, 604.31814317, 4492.95104691) km>),
'476': ('Grange Observatory, Bussoleno',
 <EarthLocation (4471.81061724, 560.22353429, 4498.86995805) km>),
'477': ('Galleywood',
 <EarthLocation (3960.87216009, 33.57045828, 4982.40928029) km>),
'478': ('Lamalou-les-Bains',
 <EarthLocation (4620.48504078, 249.39538943, 4375.21063789) km>),
'479': ('Sollies-Pont',
 <EarthLocation (4631.37149528, 490.9042871, 4343.25617152) km>),
'480': ('Cockfield',
 <EarthLocation (3920.02862697, 52.91039304, 5014.23618392) km>),
'481': ('Moorwarfen',
 <EarthLocation (3761.00101519, 523.93417018, 5107.42714363) km>),
'482': ('St. Andrews',
 <EarthLocation (3539.41800845, -174.01049654, 5285.30700642) km>),
'483': ('Carter Observatory, Black Birch Station',
 <EarthLocation (-4738.78901068, 514.49596881, -4225.77088798) km>),
'484': ('Happy Valley, Wellington',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-4775.73825559, 438.03858324, -4190.81869722) km>),
'485': ('Carter Observatory, Wellington',
 <EarthLocation (-4779.91259967, 437.91670346, -4186.29021995) km>),
'486': ('Palmerston North',
 <EarthLocation (-4864.03251647, 385.37034319, -4101.142091) km>),
'487': ('Macnairston Observatory',
 <EarthLocation (3615.02409301, -288.03859609, 5229.37074493) km>),
'488': ('Newcastle-upon-Tyne',
 <EarthLocation (3665.04564219, -104.52500184, 5201.56206761) km>),
'489': ('Hemingford Abbots',
 <EarthLocation (3903.40979653, -8.85657109, 5025.971956) km>),
'490': ('Wimborne Minster',
 <EarthLocation (4034.90126993, -140.90185717, 4923.921764) km>),
'491': ('Centro Astronomico de Yebes',
 <EarthLocation (4848.6339318, -262.59262721, 4123.08288228) km>),
'492': ('Mickleover',
 <EarthLocation (3857.39716196, -103.0306408, 5070.618915) km>),
'493': ('Calar Alto',
 <EarthLocation (5081.73515071, -225.94337464, 3838.49040934) km>),
'494': ('Stakenbridge',
 <EarthLocation (3895.91987849, -147.20790322, 5031.01068423) km>),
'495': ('Altrincham',
 <EarthLocation (3810.04085258, -155.52680123, 5095.72326223) km>),
'496': ('Bishopstoke',
 <EarthLocation (4024.18376656, -92.30531022, 4930.9377147) km>),
'497': ('Loudwater',
 <EarthLocation (3968.96044317, -48.87415505, 4975.96736192) km>),
'498': ('Earls Barton',
 <EarthLocation (3911.63859991, -50.65306563, 5020.74188366) km>),
'499': ('Cheam',
 <EarthLocation (3989.9955867, -14.56984431, 4959.39696199) km>),
'500': ('Geocentric', <EarthLocation (0., 0., 0.) km>),
'501': ('Herstmonceux',
 <EarthLocation (4033.26831276, 24.4621495, 4924.43201496) km>),
'502': ('Colchester',
 <EarthLocation (3941.96492421, 58.34206967, 4997.11088608) km>),
'503': ('Cambridge',
 <EarthLocation (3916.17075752, 6.47959291, 5017.48903379) km>),
'504': ('Le Creusot',
 <EarthLocation (4359.5504641, 335.02051117, 4628.6140209) km>),
'505': ('Simon Stevin',
 <EarthLocation (3960.34417816, 316.13043089, 4973.0334189) km>),
'506': ('Bendestorf',
 <EarthLocation (3756.6422513, 659.69354637, 5083.375189) km>),
'507': ('Nyenheim',
 <EarthLocation (3918.98962039, 358.03535435, 4994.081271) km>),
'508': ('Zeist', <EarthLocation (3918.549273, 362.82303501, 4994.081271) km>),
'509': ('La Seyne sur Mer',
 <EarthLocation (4639.98025308, 477.24461326, 4335.60240712) km>),
'510': ('Siegen',
 <EarthLocation (3990.55493078, 562.65424953, 4927.55730209) km>),
'511': ('Haute Provence',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4578.31238503, 458.24339505, 4403.08309658) km>),
'512': ('Leiden (before 1860)',
 <EarthLocation (3909.05726459, 306.91495055, 5013.59837022) km>),
'513': ('Lyons',
 <EarthLocation (4447.28877577, 372.31587288, 4541.80757633) km>),
'514': ('Mundenheim (1907-1913)',
 <EarthLocation (4109.11365851, 609.56608029, 4823.7850131) km>),
'515': ('Volkssternwarte Dhaun, near Kirn',
 <EarthLocation (4088.60948932, 537.95622942, 4849.80781206) km>),
'516': ('Hamburg (before 1909)',
 <EarthLocation (3740.15120663, 657.68654963, 5107.23579952) km>),
'517': ('Geneva (from 1967)',
 <EarthLocation (4388.44985923, 471.76352454, 4589.51604109) km>),
'518': ('Marine Observatory, Hamburg',
 <EarthLocation (3740.47743072, 657.70959035, 5107.03807727) km>),
'519': ('Meschede',
 <EarthLocation (3950.9006913, 575.43747361, 4957.66210873) km>),
'520': ('Bonn',
 <EarthLocation (4014.46977411, 499.78672558, 4914.54590261) km>),
'521': ('Remeis Observatory, Bamberg',
 <EarthLocation (4043.75030713, 777.82088712, 4854.7444901) km>),
'522': ('Strasbourg',
 <EarthLocation (4188.57592672, 571.35813564, 4760.19498721) km>),
'523': ('Frankfurt',
 <EarthLocation (4051.39104486, 616.41895363, 4871.23835238) km>),
'524': ('Mannheim',
 <EarthLocation (4106.35047479, 610.80431856, 4825.6984542) km>),
'525': ('Marburg',
 <EarthLocation (3991.24413827, 615.67832263, 4920.5604858) km>),
'526': ('Kiel',
 <EarthLocation (3668.19624725, 656.55648931, 5159.01989382) km>),
'527': ('Altona',
 <EarthLocation (3741.13083259, 655.8321724, 5106.9742959) km>),
'528': ('Gottingen',
 <EarthLocation (3916.41398623, 686.52464112, 4970.54594547) km>),
'529': ('Christiania',
 <EarthLocation (3149.61353663, 596.42953744, 5495.59418331) km>),
'530': ('Lubeck',
 <EarthLocation (3704.68954828, 699.32531126, 5127.3843343) km>),
'531': ('Collegio Romano, Rome',
 <EarthLocation (4642.24395696, 1027.43525053, 4237.25153458) km>),
'532': ('Munich',
 <EarthLocation (4176.75953952, 858.00376969, 4728.1129581) km>),
'533': ('Padua',
 <EarthLocation (4390.11221119, 922.86128598, 4518.71872039) km>),
'534': ('Leipzig (since 1861)',
 <EarthLocation (3900.07676236, 856.86668066, 4957.02429503) km>),
'535': ('Palermo',
 <EarthLocation (4888.88383021, 1160.891913, 3915.28317882) km>),
'536': ('Berlin-Babelsberg',
 <EarthLocation (3797.7035634, 884.18651886, 5030.62799601) km>),
'537': ('Urania Observatory, Berlin',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (3783.44468424, 898.84586351, 5038.72823) km>),
'538': ('Pola',
 <EarthLocation (4396.76511316, 1083.70088627, 4476.62301619) km>),
'539': ('Kremsmunster',
 <EarthLocation (4142.05027872, 1042.84002632, 4721.35213288) km>),
'540': ('Linz',
 <EarthLocation (4108.6394272, 1045.39296521, 4750.24509349) km>),
'541': ('Prague',
 <EarthLocation (3972.75149446, 1019.68185129, 4868.49575347) km>),
'542': ('Falkensee',
 <EarthLocation (3784.78165672, 876.38829503, 5041.2794848) km>),
'543': ('Leipzig (before 1861)',
 <EarthLocation (3900.03914673, 855.25308492, 4957.0880764) km>),
'544': ('Wilhelm Foerster Observatory, Berlin',
 <EarthLocation (3789.50503408, 899.38503543, 5034.02754303) km>),
'545': ('Vienna (before 1879)',
 <EarthLocation (4085.61348451, 1201.04361261, 4732.577654) km>),
'546': ('Oppolzer Observatory, Vienna',
 <EarthLocation (4085.74641998, 1199.00672308, 4733.02412359) km>),
'547': ('Breslau',
 <EarthLocation (3836.04952018, 1175.45606315, 4941.71676623) km>),
'548': ('Berlin (1835-1913)',
 <EarthLocation (3784.76029452, 901.30806514, 5037.19747712) km>),
'549': ('Uppsala',
 <EarthLocation (3060.08591886, 972.22737003, 5492.59645892) km>),
'550': ('Schwerin',
 <EarthLocation (3715.48766155, 750.49650445, 5112.0768055) km>),
'551': ("Hurbanovo, formerly O'Gyalla",
 <EarthLocation (4071.98872622, 1337.97492562, 4707.57535696) km>),
'552': ('Osservatorio S. Vittore, Bologna',
 <EarthLocation (4470.37252391, 896.6606394, 4445.561489) km>),
'553': ('Chorzow',
 <EarthLocation (3859.87849059, 1328.59556595, 4883.99462638) km>),
'554': ('Burgsolms Observatory, Wetzlar',
 <EarthLocation (4018.32101326, 593.08021029, 4901.27937765) km>),
'555': ('Cracow-Fort Skala',
 <EarthLocation (3860.20766589, 1391.76219384, 4866.90121922) km>),
'556': ('Reintal, near Munich',
 <EarthLocation (4222.37179159, 840.64809651, 4681.552558) km>),
'557': ('Ondrejov',
 <EarthLocation (3979.3286623, 1049.86913526, 4857.13629147) km>),
'558': ('Warsaw',
 <EarthLocation (3655.08454387, 1405.27370885, 5017.80794064) km>),
'559': ('Serra La Nave',
 <EarthLocation (4885.97680516, 1307.36573104, 3871.529159) km>),
'560': ('Madonna di Dossobuono',
 <EarthLocation (4404.11760527, 850.57085235, 4519.29913086) km>),
'561': ('Piszkesteto Stn. (Konkoly)',
 <EarthLocation (4027.50281479, 1457.48250697, 4711.46602053) km>),
'562': ('Figl Observatory, Vienna',
 <EarthLocation (4105.57367298, 1171.33193662, 4723.77582494) km>),
'563': ('Seewalchen',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4159.73058205, 1006.34473854, 4713.443243) km>),
'564': ('Herrsching',
 <EarthLocation (4198.36815553, 830.53782495, 4726.199517) km>),
'565': ('Bassano Bresciano',
 <EarthLocation (4422.4740202, 790.50237073, 4512.27680202) km>),
'566': ('Haleakala-NEAT/GEODSS',
 <EarthLocation (-5466.01285689, -2404.23785578, 2242.29784372) km>),
'567': ('Chions',
 <EarthLocation (4341.7632325, 979.39027698, 4553.16066019) km>),
'568': ('Mauna Kea',
 <EarthLocation (-5464.34160705, -2493.44659532, 2151.02670325) km>),
'569': ('Helsinki',
 <EarthLocation (2884.94543524, 1342.74129214, 5509.11583375) km>),
'570': ('Vilnius (since 1939)',
 <EarthLocation (3341.05791543, 1579.23949999, 5180.9606851) km>),
'571': ('Cavriana',
 <EarthLocation (4413.15152154, 828.29074271, 4515.720996) km>),
'572': ('Cologne',
 <EarthLocation (3995.5399395, 482.80590991, 4923.921764) km>),
'573': ('Eldagsen',
 <EarthLocation (3863.77817052, 657.75505633, 5014.4913094) km>),
'574': ('Gottolengo',
 <EarthLocation (4422.26867014, 801.00762118, 4510.68226777) km>),
'575': ('La Chaux de Fonds',
 <EarthLocation (4320.42151747, 515.79090773, 4649.27918478) km>),
'576': ('Burwash',
 <EarthLocation (4022.41304644, 26.39721512, 4933.23384402) km>),
'577': ('Metzerlen Observatory',
 <EarthLocation (4287.38124227, 564.44444865, 4681.552558) km>),
'578': ('Linden Observatory',
 <EarthLocation (5057.61074831, 2688.047164, -2800.002143) km>),
'579': ('Novi Ligure',
 <EarthLocation (4480.86568526, 697.679205, 4471.074037) km>),
'580': ('Graz',
 <EarthLocation (4194.39721839, 1162.7047599, 4647.23818094) km>),
'581': ('Sedgefield',
 <EarthLocation (4880.20866515, 2051.45082502, -3546.244172) km>),
'582': ('Orwell Park',
 <EarthLocation (3933.23997018, 85.19174148, 5003.45713239) km>),
'583': ('Odessa-Mayaki',
 <EarthLocation (3805.61861613, 2221.30313249, 4595.83039672) km>),
'584': ('Leningrad',
 <EarthLocation (2765.30943072, 1615.56676102, 5497.25249893) km>),
'585': ('Kyiv comet station',
 <EarthLocation (3516.72032508, 2073.54255661, 4884.14132353) km>),
'586': ('Pic du Midi',
 <EarthLocation (4678.85931015, 11.62045537, 4324.31310463) km>),
'587': ('Sormano',
 <EarthLocation (4390.90208029, 713.46603548, 4557.04494562) km>),
'588': ('Eremo di Tizzano',
 <EarthLocation (4472.74176349, 889.68365285, 4445.561489) km>),
'589': ('Santa Lucia Stroncone',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4594.30725313, 1030.62832657, 4288.57002488) km>),
'590': ('Metzerlen',
 <EarthLocation (4287.77425399, 561.45115459, 4681.552558) km>),
'591': ('Resse Observatory',
 <EarthLocation (3835.57188591, 650.5151083, 5037.38882123) km>),
'592': ('Solingen',
 <EarthLocation (3976.98441775, 489.80183349, 4945.84342087) km>),
'593': ('Monte Argentario',
 <EarthLocation (4624.15517611, 913.0915136, 4279.729927) km>),
'594': ('Monte Autore',
 <EarthLocation (4625.91783044, 1085.2803265, 4243.31076473) km>),
'595': ("Farra d'Isonzo",
 <EarthLocation (4321.79778887, 1039.62872864, 4558.76704261) km>),
'596': ('Colleverde di Guidonia',
 <EarthLocation (4633.58313873, 1037.28254175, 4244.33126665) km>),
'597': ('Springe',
 <EarthLocation (3864.4479971, 658.00094962, 5014.55509077) km>),
'598': ('Loiano',
 <EarthLocation (4486.71536981, 899.31063553, 4429.28448338) km>),
'599': ('Campo Imperatore-CINEOS',
 <EarthLocation (4584.02949668, 1105.40666317, 4283.58232175) km>),
'600': ('TLC Observatory, Bologna',
 <EarthLocation (4476.65587225, 908.41027509, 4436.88722268) km>),
'601': ('Engelhardt Observatory, Dresden',
 <EarthLocation (3903.99466549, 953.72000257, 4936.29534978) km>),
'602': ('Urania Observatory, Vienna',
 <EarthLocation (4085.35234314, 1201.25346992, 4732.76899811) km>),
'603': ('Bothkamp',
 <EarthLocation (3680.70516871, 657.62199613, 5150.02672065) km>),
'604': ('Archenhold Sternwarte, Berlin-Treptow',
 <EarthLocation (3785.01396972, 906.97169064, 5035.99838736) km>),
'605': ('Marl',
 <EarthLocation (3932.99826101, 490.78706885, 4980.43205782) km>),
'606': ('Norderstedt',
 <EarthLocation (3728.15411663, 657.07895985, 5116.03125044) km>),
'607': ('Hagen Observatory, Ronkhausen',
 <EarthLocation (3964.59425733, 557.18738618, 4949.434312) km>),
'608': ('Haleakala-AMOS',
 <EarthLocation (-5466.02964151, -2404.19969575, 2242.29784372) km>),
'609': ('Osservatorio Polino',
 <EarthLocation (4587.37826769, 1046.71545282, 4293.37914018) km>),
'610': ('Pianoro',
 <EarthLocation (4476.1054467, 897.9161865, 4439.43847748) km>),
'611': ('Starkenburg Sternwarte, Heppenheim',
 <EarthLocation (4090.84326326, 622.56040539, 4837.68935176) km>),
'612': ('Lenkerbeck',
 <EarthLocation (3955.76815582, 492.71697003, 4962.190586) km>),
'613': ('Heisingen',
 <EarthLocation (3956.27107717, 490.73921357, 4962.190586) km>),
'614': ('Soisy-sur-Seine',
 <EarthLocation (4219.05057483, 181.77315629, 4763.8305253) km>),
'615': ('St. Veran',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4510.3686255, 546.34973761, 4465.58883918) km>),
'616': ('Brno',
 <EarthLocation (4001.79982759, 1191.73207503, 4805.70937284) km>),
'617': ('Arbonne la Foret',
 <EarthLocation (4236.93180833, 190.36021723, 4747.69383869) km>),
'618': ('Martigues',
 <EarthLocation (4622.38331574, 405.03210454, 4361.49764334) km>),
'619': ('Sabadell',
 <EarthLocation (4777.27147425, 174.35053946, 4208.47338044) km>),
'620': ('Observatorio Astronomico de Mallorca',
 <EarthLocation (4911.65648356, 253.25732207, 4047.75708431) km>),
'621': ('Bergisch Gladbach',
 <EarthLocation (3980.57820384, 522.99504386, 4939.87348464) km>),
'622': ('Oberwichtrach',
 <EarthLocation (4348.543146, 577.74779553, 4615.09237046) km>),
'623': ('Liege',
 <EarthLocation (4035.90443094, 393.35582705, 4906.82835684) km>),
'624': ('Dertingen',
 <EarthLocation (4070.11071553, 689.62801166, 4845.91714849) km>),
'625': ('Kihei-AMOS Remote Maui Experimental Site',
 <EarthLocation (-5469.43474697, -2385.93449268, 2245.16800537) km>),
'626': ('Geel',
 <EarthLocation (3993.29714387, 348.41312667, 4944.58692788) km>),
'627': ('Blauvac',
 <EarthLocation (4573.37956292, 417.38520411, 4411.62980016) km>),
'628': ('Mulheim-Ruhr',
 <EarthLocation (3956.5967645, 474.85371734, 4963.36416321) km>),
'629': ('Szeged Observatory',
 <EarthLocation (4147.86948716, 1522.10079101, 4584.6048756) km>),
'630': ('Osenbach',
 <EarthLocation (4242.5381369, 538.71829623, 4716.69609287) km>),
'631': ('Hamburg-Georgswerder',
 <EarthLocation (3743.30358418, 661.59019029, 5104.46768806) km>),
'632': ('San Polo A Mosciano',
 <EarthLocation (4528.98311634, 894.6190211, 4386.43615901) km>),
'633': ('Romito',
 <EarthLocation (4519.01111777, 791.44917147, 4416.09449606) km>),
'634': ('Crolles',
 <EarthLocation (4458.26461117, 401.46623281, 4528.92373959) km>),
'635': ('Pergignan',
 <EarthLocation (4688.60771869, 237.67050012, 4303.13768979) km>),
'636': ('Essen',
 <EarthLocation (3958.31590569, 484.57552412, 4961.10630271) km>),
'637': ('Hamburg-Himmelsmoor',
 <EarthLocation (3725.36752599, 662.9383411, 5117.30687784) km>),
'638': ('Detmold',
 <EarthLocation (3892.9151684, 609.14786621, 4998.80109238) km>),
'639': ('Dresden',
 <EarthLocation (3899.36554593, 952.24297839, 4940.24979472) km>),
'640': ('Senftenberger Sternwarte',
 <EarthLocation (3985.53697154, 964.17341628, 4869.0697858) km>),
'641': ('Overberg',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4941.85531413, 1801.34553124, -3595.61095238) km>),
'642': ('Oak Bay, Victoria',
 <EarthLocation (-2328.88631129, -3543.36865617, 4748.5229965) km>),
'643': ('OCA-Anza Observatory',
 <EarthLocation (-2394.8693678, -4757.40578858, 3499.55620916) km>),
'644': ('Palomar Mountain/NEAT',
 <EarthLocation (-2410.03526915, -4758.71525202, 3488.05642815) km>),
'645': ('Apache Point-Sloan Digital Sky Survey',
 <EarthLocation (-1464.01185705, -5166.62412611, 3435.02859713) km>),
'646': ('Santana Observatory, Rancho Cucamonga',
 <EarthLocation (-2449.5700061, -4692.95414357, 3546.43551611) km>),
'647': ('Stone Finder Observatory, Calgary',
 <EarthLocation (-1646.00113724, -3691.48113322, 4918.8192544) km>),
'648': ('Winer Observatory, Sonoita',
 <EarthLocation (-1912.38440826, -5087.33637416, 3329.72555526) km>),
'649': ('Powell Observatory, Louisburg',
 <EarthLocation (-408.69550927, -4971.37852597, 3961.90736029) km>),
'650': ('Temecula',
 <EarthLocation (-2426.1745673, -4741.73221536, 3497.51520532) km>),
'651': ('Grasslands Observatory, Tucson',
 <EarthLocation (-1910.42186537, -5087.7602882, 3330.17202485) km>),
'652': ('Rock Finder Observatory, Calgary',
 <EarthLocation (-1636.29121205, -3663.68596673, 4942.4183613) km>),
'653': ('Torus Observatory, Buckley',
 <EarthLocation (-2309.89791069, -3677.69978751, 4655.78488452) km>),
'654': ('Table Mountain Observatory, Wrightwood-PHMC',
 <EarthLocation (-2448.84314313, -4668.00258764, 3582.7717626) km>),
'655': ('Sooke',
 <EarthLocation (-2350.35554231, -3535.29332857, 4744.0583006) km>),
'656': ('Victoria',
 <EarthLocation (-2350.50008775, -3536.72897892, 4743.22914279) km>),
'657': ('Climenhaga Observatory, Victoria',
 <EarthLocation (-2327.05239361, -3541.29203929, 4751.13803267) km>),
'658': ('Dominion Astrophysical Observatory',
 <EarthLocation (-2331.08439511, -3532.99363381, 4755.54532534) km>),
'659': ('Heron Cove Observatory, Orcas',
 <EarthLocation (-2298.44357609, -3546.25350399, 4761.2792705) km>),
'660': ('Leuschner Observatory, Berkeley',
 <EarthLocation (-2690.93215696, -4262.87424491, 3894.42667083) km>),
'661': ('Rothney Astrophysical Observatory, Priddis',
 <EarthLocation (-1659.39495377, -3677.15432411, 4925.32495414) km>),
'662': ('Lick Observatory, Mount Hamilton',
 <EarthLocation (-2664.34383693, -4323.13283749, 3848.24895895) km>),
'663': ('Red Mountain Observatory',
 <EarthLocation (-1967.60143608, -4947.7823208, 3500.25780423) km>),
'664': ('Manastash Ridge Observatory',
 <EarthLocation (-2228.7908027, -3750.35586729, 4638.8190401) km>),
'665': ('Wallis Observatory',
 <EarthLocation (-2543.01398985, -4585.88753485, 3621.56996997) km>),
'666': ('Moorpark College Observatory',
 <EarthLocation (-2543.59978979, -4620.9050837, 3574.3079748) km>),
'667': ('Wanapum Dam',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-2182.25539705, -3781.1807586, 4634.52017576) km>),
'668': ('San Emigdio Peak',
 <EarthLocation (-2553.05720458, -4571.90469151, 3622.781816) km>),
'669': ('Ojai',
 <EarthLocation (-2566.43499887, -4596.5772047, 3589.55172223) km>),
'670': ('Camarillo',
 <EarthLocation (-2563.11518267, -4615.58137724, 3566.78177314) km>),
'671': ('Stony Ridge',
 <EarthLocation (-2476.72069164, -4658.42250733, 3575.09248565) km>),
'672': ('Mount Wilson',
 <EarthLocation (-2483.73390138, -4660.15173085, 3568.05740054) km>),
'673': ('Table Mountain Observatory, Wrightwood',
 <EarthLocation (-2448.89928598, -4667.99474223, 3582.73987191) km>),
'674': ('Ford Observatory, Wrightwood',
 <EarthLocation (-2447.38513185, -4668.71676343, 3582.79089701) km>),
'675': ('Palomar Mountain',
 <EarthLocation (-2410.35672192, -4758.78122903, 3487.76303385) km>),
'676': ('San Clemente',
 <EarthLocation (-2469.72892777, -4722.2886653, 3492.79538394) km>),
'677': ('Lake Arrowhead',
 <EarthLocation (-2410.10193761, -4695.21377504, 3572.52209644) km>),
'678': ('Fountain Hills',
 <EarthLocation (-1969.1602775, -4938.16808343, 3513.26920371) km>),
'679': ('San Pedro Martir',
 <EarthLocation (-2352.56312783, -4940.39257825, 3271.47403004) km>),
'680': ('Los Angeles',
 <EarthLocation (-2263.83825701, -4806.54544553, 3533.487898) km>),
'681': ('Calgary',
 <EarthLocation (-1640.4745255, -3664.88964684, 4940.44113883) km>),
'682': ('Kanab',
 <EarthLocation (-1939.62773638, -4714.78597032, 3822.54506684) km>),
'683': ('Goodricke-Pigott Observatory, Tucson',
 <EarthLocation (-1944.3732636, -5043.73076666, 3375.43766314) km>),
'684': ('Prescott',
 <EarthLocation (-2013.11033634, -4862.47848335, 3594.46288772) km>),
'685': ('Williams',
 <EarthLocation (-1963.1303666, -4820.12051636, 3667.428775) km>),
'686': ('U. of Minn. Infrared Obs., Mt. Lemmon',
 <EarthLocation (-1913.32077816, -5039.2898497, 3403.31012183) km>),
'687': ('Northern Arizona University, Flagstaff',
 <EarthLocation (-1926.21279998, -4852.01465795, 3655.82056566) km>),
'688': ('Lowell Observatory, Anderson Mesa Station',
 <EarthLocation (-1918.39099983, -4861.2842156, 3647.84789441) km>),
'689': ('U.S. Naval Observatory, Flagstaff',
 <EarthLocation (-1933.66585161, -4849.25521907, 3655.88434703) km>),
'690': ('Lowell Observatory, Flagstaff',
 <EarthLocation (-1926.73369365, -4850.70979304, 3657.47888128) km>),
'691': ('Steward Observatory, Kitt Peak-Spacewatch',
 <EarthLocation (-1994.53259422, -5037.52695059, 3357.95518962) km>),
'692': ('Steward Observatory, Tucson',
 <EarthLocation (-1931.0093945, -5043.94528255, 3382.70873932) km>),
'693': ('Catalina Station, Tucson',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-1908.64138094, -5042.40755724, 3400.89280791) km>),
'694': ('Tumamoc Hill, Tucson',
 <EarthLocation (-1936.50647048, -5043.27214403, 3380.98664233) km>),
'695': ('Kitt Peak',
 <EarthLocation (-1994.12327103, -5037.94966269, 3357.61077023) km>),
'696': ('Whipple Observatory, Mt. Hopkins',
 <EarthLocation (-1937.32502184, -5077.44729618, 3332.51280113) km>),
'697': ('Kitt Peak, McGraw-Hill',
 <EarthLocation (-1996.11404253, -5037.44936719, 3356.8645282) km>),
'698': ('Mt. Bigelow',
 <EarthLocation (-1908.64704363, -5042.39859404, 3400.89918604) km>),
'699': ('Lowell Observatory-LONEOS',
 <EarthLocation (-1918.49196546, -4861.24437069, 3647.84789441) km>),
'700': ('Chinle',
 <EarthLocation (-1727.22813148, -4845.72200432, 3760.5495752) km>),
'701': ('Junk Bond Observatory, Sierra Vista',
 <EarthLocation (-1880.67828426, -5110.75215297, 3311.68180569) km>),
'702': ('Joint Obs. for cometary research, Socorro',
 <EarthLocation (-1565.34485785, -5060.47010398, 3546.8819857) km>),
'703': ('Catalina Sky Survey',
 <EarthLocation (-1908.64478572, -5042.39262893, 3400.90556418) km>),
'704': ('Lincoln Laboratory ETS, New Mexico',
 <EarthLocation (-1521.03932783, -5083.07799043, 3530.56671125) km>),
'705': ('Apache Point',
 <EarthLocation (-1464.01185705, -5166.62412611, 3435.02859713) km>),
'706': ('Salida',
 <EarthLocation (-1381.76069325, -4798.7251351, 3957.18753891) km>),
'707': ('Chamberlin field station',
 <EarthLocation (-1314.2874413, -4758.51222259, 4037.360721) km>),
'708': ('Chamberlin Observatory, Denver',
 <EarthLocation (-1268.68397032, -4750.54293787, 4051.3926224) km>),
'709': ('W & B Observatory, Cloudcroft',
 <EarthLocation (-1456.61831494, -5157.48050339, 3451.18441815) km>),
'710': ('MPO Observatory, Florissant',
 <EarthLocation (-1309.60423486, -4798.16030141, 3983.65680746) km>),
'711': ('McDonald Observatory, Fort Davis',
 <EarthLocation (-1330.74819784, -5328.8201142, 3235.69268147) km>),
'712': ('USAF Academy Observatory, Colorado Springs',
 <EarthLocation (-1274.97715847, -4798.007736, 3994.30829625) km>),
'713': ('Thornton',
 <EarthLocation (-1269.72588096, -4735.27631346, 4068.80493641) km>),
'714': ('Bagdad',
 <EarthLocation (-2070.04126539, -4833.79849496, 3599.75674143) km>),
'715': ('Jornada Observatory, Las Cruces',
 <EarthLocation (-1551.75242067, -5164.36696291, 3397.25089168) km>),
'716': ('Palmer Divide Observatory, Colorado Springs',
 <EarthLocation (-1262.71223598, -4795.74307566, 4001.06912147) km>),
'717': ('Prude Ranch',
 <EarthLocation (-1324.98143276, -5333.28009465, 3229.63345132) km>),
'718': ('Tooele',
 <EarthLocation (-1839.13779218, -4485.21776904, 4133.16033874) km>),
'719': ('Etscorn Observatory',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-1539.02237517, -5061.09813, 3553.92344895) km>),
'720': ('Universidad de Monterrey',
 <EarthLocation (-1036.1470111, -5660.04125913, 2743.74697466) km>),
'721': ('Lime Creek',
 <EarthLocation (-867.20710996, -4786.74785794, 4112.36761212) km>),
'722': ('Missouri City',
 <EarthLocation (-539.74028937, -5523.7564191, 3132.3030807) km>),
'723': ('Cottonwood Observatory, Ada',
 <EarthLocation (-608.46925948, -5203.16198518, 3626.67247957) km>),
'724': ('National Observatory, Tacubaya',
 <EarthLocation (-961.96635547, -5942.8427563, 2106.44352562) km>),
'725': ('Fair Oaks Ranch',
 <EarthLocation (-833.88320994, -5478.41645927, 3148.12086046) km>),
'726': ('Brainerd',
 <EarthLocation (-330.60309986, -4390.01435751, 4599.9124044) km>),
'727': ('Zeno Observatory, Edmond',
 <EarthLocation (-674.13876304, -5147.4705811, 3693.55362415) km>),
'728': ('Corpus Christi',
 <EarthLocation (-727.08816371, -5604.70203429, 2946.31660578) km>),
'729': ('Glenlea Astronomical Observatory, Winnipeg',
 <EarthLocation (-513.01793644, -4106.23761955, 4837.50438579) km>),
'730': ('University of North Dakota, Grand Forks',
 <EarthLocation (-533.75440342, -4249.8123885, 4710.49016557) km>),
'731': ('Rose-Hulman Observatory, Terre Haute',
 <EarthLocation (229.73509273, -4924.30605082, 4033.78896428) km>),
'732': ('Oaxaca',
 <EarthLocation (-718.73012259, -6054.41332673, 1872.55724183) km>),
'733': ('Allen, Texas',
 <EarthLocation (-618.54681331, -5309.09530161, 3468.87737019) km>),
'734': ('Farpoint Observatory, Eskridge',
 <EarthLocation (-519.76433936, -4944.06525957, 3983.08277513) km>),
'735': ('George Observatory, Needville',
 <EarthLocation (-542.19466857, -5536.09636715, 3110.19645786) km>),
'736': ('Houston',
 <EarthLocation (-644.01391871, -5511.86568486, 3133.70627084) km>),
'737': ('New Bullpen Observatory, Alpharetta',
 <EarthLocation (521.27704379, -5256.58971431, 3562.8273282) km>),
'738': ('Observatory of the State University of Missouri',
 <EarthLocation (-201.65926248, -4963.19798512, 3987.6112524) km>),
'739': ('Sunflower Observatory, Olathe',
 <EarthLocation (-412.30052591, -4955.59258764, 3981.16933403) km>),
'740': ('SFA Observatory, Nacogdoches',
 <EarthLocation (-441.18106199, -5410.47480352, 3337.6790921) km>),
'741': ('Goodsell Observatory, Northfield',
 <EarthLocation (-250.54466639, -4553.03320032, 4445.05123804) km>),
'742': ('Drake University, Des Moines',
 <EarthLocation (-307.26904489, -4767.26235974, 4212.30026264) km>),
'743': ('University of Minnesota, Minneapolis',
 <EarthLocation (-176.51805581, -4515.74840197, 4486.26675933) km>),
'744': ('Doyan Rose Observatory, Indianapolis',
 <EarthLocation (328.21975622, -4892.77029109, 4065.10561695) km>),
'745': ('Morrison Obervatory, Glasgow',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-243.86414832, -4941.44330433, 4012.23086122) km>),
'746': ('Brooks Observatory, Mt. Pleasant',
 <EarthLocation (421.43630519, -4608.17125644, 4375.21063789) km>),
'747': ('Highland Road Park Observatory',
 <EarthLocation (-102.95140928, -5508.0262143, 3203.54687099) km>),
'748': ('Van Allen Observatory, Iowa City',
 <EarthLocation (-127.91472747, -4782.78546689, 4203.95765944) km>),
'749': ('Oakwood',
 <EarthLocation (567.04028939, -5250.24639271, 3565.50614574) km>),
'750': ('Hobbs Observatory, Fall Creek',
 <EarthLocation (-100.59431994, -4531.12387403, 4473.05125947) km>),
'751': ('Lake Saint Louis',
 <EarthLocation (-65.68163794, -4976.93716405, 3975.11010388) km>),
'752': ('Puckett Observatory, Mountain Town',
 <EarthLocation (499.76129282, -5223.95260924, 3613.78864283) km>),
'753': ('Washburn Observatory, Madison',
 <EarthLocation (48.10717923, -4666.13736589, 4333.88668827) km>),
'754': ('Yerkes Observatory, Williams Bay',
 <EarthLocation (118.49076134, -4703.14902733, 4292.67754511) km>),
'755': ('Optec Observatory',
 <EarthLocation (379.10552717, -4663.16998729, 4320.74134791) km>),
'756': ('Dearborn Observatory, Evanston (bef. July 1939)',
 <EarthLocation (192.46082342, -4738.90797746, 4250.43514376) km>),
'757': ('High Point',
 <EarthLocation (897.11776342, -5085.21263714, 3731.8479587) km>),
'758': ('BCC Observatory, Cocoa',
 <EarthLocation (901.49011588, -5542.73467088, 3014.11620209) km>),
'759': ('Nashville',
 <EarthLocation (287.72104453, -5154.82331736, 3733.1235861) km>),
'760': ('Goethe Link Observatory, Brooklyn',
 <EarthLocation (309.65145929, -4915.1980933, 4039.72063169) km>),
'761': ('Zephyrhills',
 <EarthLocation (747.92204492, -5571.58651632, 3003.01824371) km>),
'762': ('Four Winds Observatory, Lake Leelanau',
 <EarthLocation (331.18327674, -4508.99615778, 4484.08543648) km>),
'763': ('King City',
 <EarthLocation (835.9962568, -4525.70665431, 4401.48856233) km>),
'764': ('Puckett Observatory, Stone Mountain',
 <EarthLocation (476.14303217, -5289.3040415, 3521.05053085) km>),
'765': ('Cincinnati',
 <EarthLocation (481.47348702, -4930.38200679, 4004.44953408) km>),
'766': ('Michigan State University Obs., East Lansing',
 <EarthLocation (451.29149441, -4672.56582589, 4303.77550349) km>),
'767': ('Ann Arbor',
 <EarthLocation (516.15582188, -4698.05819805, 4268.8870941) km>),
'768': ('Dearborn Observatory, Evanston (aft. Oct. 1939)',
 <EarthLocation (192.39030819, -4738.78317294, 4250.58184091) km>),
'769': ('McMillin Observatory, Columbus',
 <EarthLocation (595.39752915, -4856.69183249, 4077.92567232) km>),
'770': ('Crescent Moon Observatory, Columbus',
 <EarthLocation (351.90551061, -4935.18172656, 4011.848173) km>),
'771': ('Boyeros Observatory, Havana',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (774.70036946, -5829.39047952, 2481.095293) km>),
'772': ('Boltwood Observatory, Stittsville',
 <EarthLocation (1094.67230024, -4362.42315629, 4507.36563653) km>),
'773': ('Warner and Swasey Observatory, Cleveland',
 <EarthLocation (701.11157622, -4729.75215081, 4207.40185342) km>),
'774': ('Warner and Swasey Nassau Station, Chardon',
 <EarthLocation (741.19503088, -4719.69829653, 4212.05789343) km>),
'775': ('Sayre Observatory, South Bethlehem',
 <EarthLocation (1223.71915052, -4692.28938698, 4129.39723791) km>),
'776': ('Foggy Bottom, Hamilton',
 <EarthLocation (1170.69578887, -4537.55711947, 4312.83245803) km>),
'777': ('Toronto',
 <EarthLocation (850.21358503, -4542.33073064, 4381.46121215) km>),
'778': ('Allegheny Observatory, Pittsburgh',
 <EarthLocation (841.79051014, -4784.8717158, 4119.12843734) km>),
'779': ('David Dunlap Observatory, Richmond Hill',
 <EarthLocation (845.57518608, -4527.94959882, 4397.27899191) km>),
'780': ('Leander McCormick Observatory, Charlottesville',
 <EarthLocation (1000.97232617, -4929.71235821, 3908.5223536) km>),
'781': ('Quito',
 <EarthLocation (1272.98670997, -6252.73997806, -25.83145485) km>),
'782': ('Quito, comet astrograph station',
 <EarthLocation (1286.66637628, -6240.49682673, 0.) km>),
'783': ('Rixeyville',
 <EarthLocation (1040.03298658, -4884.58587069, 3967.201214) km>),
'784': ('Stull Observatory, Alfred University',
 <EarthLocation (1000.48001984, -4621.70137414, 4266.65474615) km>),
'785': ('Fitz-Randolph Observatory, Princeton',
 <EarthLocation (1288.97135536, -4694.23430366, 4107.32888389) km>),
'786': ('U.S. Naval Obs., Washington (since 1893)',
 <EarthLocation (1112.23526069, -4842.87216969, 3985.50646719) km>),
'787': ('U.S. Naval Obs., Washington (before 1893)',
 <EarthLocation (1113.89482994, -4844.32322516, 3983.21033787) km>),
'788': ('Mount Cuba Observatory, Wilmington',
 <EarthLocation (1217.84850072, -4754.67725978, 4059.6842005) km>),
'789': ('Litchfield Observatory, Clinton',
 <EarthLocation (1176.19449851, -4517.41951291, 4332.15821314) km>),
'790': ('Dominion Observatory, Ottawa',
 <EarthLocation (1106.927026, -4347.87868219, 4518.2722508) km>),
'791': ('Flower and Cook Observatory, Philadelphia',
 <EarthLocation (1227.02542594, -4736.50608582, 4077.98945369) km>),
'792': ('University of Rhode Island, Quonochontaug',
 <EarthLocation (1508.02323655, -4559.84102307, 4190.436009) km>),
'793': ('Dudley Observatory, Albany (before 1893)',
 <EarthLocation (1312.31284136, -4511.13225204, 4299.31080759) km>),
'794': ('Vassar College Observatory, Poughkeepsie',
 <EarthLocation (1323.63094783, -4582.82481653, 4219.83922057) km>),
'795': ('Rutherford',
 <EarthLocation (1335.18510776, -4652.57399107, 4140.0487267) km>),
'796': ('Stamford',
 <EarthLocation (1363.6442491, -4618.3819011, 4171.301598) km>),
'797': ('Yale Observatory, New Haven',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (1408.66764602, -4586.03638546, 4188.90525612) km>),
'798': ('Yale Observatory, Bethany',
 <EarthLocation (1401.55537589, -4579.87799661, 4198.21733614) km>),
'799': ('Winchester',
 <EarthLocation (1523.95954592, -4460.01003738, 4282.9189955) km>),
'800': ('Harvard Observatory, Arequipa',
 <EarthLocation (1938.02474565, -5808.61573135, -1787.21776877) km>),
'801': ('Oak Ridge Observatory',
 <EarthLocation (1489.81439558, -4467.52466224, 4287.27526307) km>),
'802': ('Harvard Observatory, Cambridge',
 <EarthLocation (1526.21345383, -4464.91510502, 4277.01284064) km>),
'803': ('Taunton',
 <EarthLocation (1541.36168119, -4497.67087712, 4237.37909732) km>),
'804': ('Santiago-San Bernardo',
 <EarthLocation (1759.63026721, -5021.31557613, -3506.44459712) km>),
'805': ('Santiago-Cerro El Roble',
 <EarthLocation (1742.44474857, -5066.17110661, -3453.44227865) km>),
'806': ('Santiago-Cerro Calan',
 <EarthLocation (1775.28638623, -5026.82873211, -3491.26463106) km>),
'807': ('Cerro Tololo Observatory, La Serena',
 <EarthLocation (1815.10803036, -5214.0089711, -3187.7928726) km>),
'808': ('El Leoncito',
 <EarthLocation (1915.95593971, -5078.25581901, -3343.03672718) km>),
'809': ('European Southern Observatory, La Silla',
 <EarthLocation (1838.1727099, -5258.92294357, -3100.10624512) km>),
'810': ('Wallace Observatory, Westford',
 <EarthLocation (1492.99102554, -4458.09734686, 4295.80283224) km>),
'811': ('Maria Mitchell Observatory, Nantucket',
 <EarthLocation (1633.51205, -4513.59788525, 4185.90115359) km>),
'812': ('Vina del Mar',
 <EarthLocation (1695.78799647, -5081.64240231, -3450.12564741) km>),
'813': ('Santiago-Quinta Normal (1862-1920)',
 <EarthLocation (1761.65924773, -5028.17398087, -3495.53798285) km>),
'814': ('North Scituate',
 <EarthLocation (1503.41118548, -4514.37726029, 4233.38638356) km>),
'815': ('Santiago-Santa Lucia (1849-1861)',
 <EarthLocation (1765.26083694, -5027.31628419, -3495.15529463) km>),
'816': ('Rand Observatory',
 <EarthLocation (1241.0157905, -4397.87136162, 4435.48403254) km>),
'817': ('Sudbury',
 <EarthLocation (1506.60929321, -4474.11230693, 4274.43607329) km>),
'818': ('Gemeaux Observatory, Laval',
 <EarthLocation (1269.02745357, -4307.15001231, 4515.0831823) km>),
'819': ('Val-des-Bois',
 <EarthLocation (1104.75577843, -4307.42092322, 4557.24266787) km>),
'820': ('Tarija',
 <EarthLocation (2543.38017018, -5362.18774137, -2333.58174046) km>),
'821': ('Cordoba-Bosque Alegre',
 <EarthLocation (2337.11135484, -4910.78964985, -3323.21347738) km>),
'822': ('Cordoba',
 <EarthLocation (2371.35370274, -4905.0814217, -3305.94786053) km>),
'823': ('Fitchburg',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (1466.07850936, -4467.22141693, 4295.93039498) km>),
'824': ('Lake Clear',
 <EarthLocation (1240.52435607, -4397.74492685, 4435.73915802) km>),
'825': ('Granville',
 <EarthLocation (1479.4798947, -4484.16390767, 4273.54313411) km>),
'826': ('Plessissville',
 <EarthLocation (1382.84150799, -4198.96994429, 4582.24496491) km>),
'827': ('Saint-Felicien',
 <EarthLocation (1272.24772572, -4025.4245399, 4765.1061527) km>),
'828': ('Assonet',
 <EarthLocation (1548.34374823, -4502.89420778, 4229.3426447) km>),
'829': ('Complejo Astronomico El Leoncito',
 <EarthLocation (1918.44783191, -5077.58769298, -3342.84538307) km>),
'830': ('Hudson',
 <EarthLocation (1492.72647106, -4443.31859826, 4310.9827983) km>),
'831': ('Rosemary Hill Observatory, University of Florida',
 <EarthLocation (717.54286381, -5514.67575726, 3112.78598148) km>),
'832': ('Etters',
 <EarthLocation (1113.3794162, -4752.51354227, 4092.2126992) km>),
'833': ('Obs. Astronomico de Mercedes, Buenos Aires',
 <EarthLocation (2742.26582537, -4481.41187236, -3604.15765596) km>),
'834': ('Buenos Aires-AAAA',
 <EarthLocation (2751.08195191, -4477.87670571, -3601.92530801) km>),
'835': ('Drum Hill Station, Chelmsford',
 <EarthLocation (1502.83896011, -4454.58528514, 4295.93039498) km>),
'836': ('Furnace Brook Observatory, Cranston',
 <EarthLocation (1512.0366476, -4518.71288522, 4225.64332524) km>),
'837': ('Jupiter',
 <EarthLocation (964.30109695, -5608.79322376, 2869.90652452) km>),
'838': ('Dayton',
 <EarthLocation (501.65930841, -4885.47688649, 4056.24000652) km>),
'839': ('La Plata',
 <EarthLocation (2780.04735214, -4437.31297371, -3629.54264122) km>),
'840': ('Flint',
 <EarthLocation (511.21884456, -4642.96928414, 4328.8415819) km>),
'841': ('Martin Observatory, Blacksburg',
 <EarthLocation (833.14532801, -5009.65077807, 3847.41980114) km>),
'842': ('Gettysburg College Observatory',
 <EarthLocation (1083.97422256, -4783.57236123, 4063.70242681) km>),
'843': ('Emerald Lane Observatory, Decatur',
 <EarthLocation (281.26788162, -5253.22675561, 3594.52666909) km>),
'844': ('Observatorio Astronomico Los Molinos',
 <EarthLocation (2919.08020009, -4358.85367854, -3615.66381511) km>),
'845': ('Ford Observatory, Ithaca',
 <EarthLocation (1101.4210427, -4585.70376023, 4280.17639659) km>),
'846': ('Principia Astronomical Observatory, Elsah',
 <EarthLocation (-29.94895786, -4973.84894958, 3979.48550586) km>),
'847': ('Lunar Cafe Observator, Flint',
 <EarthLocation (469.9289844, -4489.92023712, 4490.8462617) km>),
'848': ('Tenagra Observatory, Cottage Grove',
 <EarthLocation (-2513.95458372, -3874.39186816, 4384.39515517) km>),
'849': ('Everstar Observatory, Olathe',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-418.5483344, -4952.63648242, 3984.23083979) km>),
'850': ('Cordell-Lorenz Observatory, Sewanee',
 <EarthLocation (371.27176416, -5204.72861615, 3656.77728621) km>),
'851': ('Burke-Gaffney Observatory, Halifax',
 <EarthLocation (2022.94042467, -4071.68557435, 4458.22846908) km>),
'852': ('River Moss Observatory, St. Peters',
 <EarthLocation (-51.69556405, -4977.86750439, 3974.2171647) km>),
'853': ('Biosphere 2 Observatory',
 <EarthLocation (-1915.04024417, -5028.60518576, 3415.10967528) km>),
'854': ('Sabino Canyon Observatory, Tucson',
 <EarthLocation (-1918.30295923, -5044.64966342, 3389.02947309) km>),
'855': ('Wayside Observatory, Minnetonka',
 <EarthLocation (-273.16581317, -4515.75798833, 4481.2790562) km>),
'856': ('Riverside',
 <EarthLocation (-2440.00292601, -4698.00732481, 3546.244172) km>),
'857': ('Iowa Robotic Observatory, Sonoita',
 <EarthLocation (-1912.33105569, -5087.45863754, 3329.64263948) km>),
'858': ('Tebbutt Observatory, Edgewood',
 <EarthLocation (-1459.82777927, -5018.22124264, 3647.6565503) km>),
'859': ('Wykrota Observatory-CEAMIG',
 <EarthLocation (4341.30143101, -4147.23183534, -2149.87863859) km>),
'860': ('Valinhos',
 <EarthLocation (4009.18789977, -4294.11073026, -2477.39597354) km>),
'861': ('Barao Geraldo',
 <EarthLocation (4006.91147929, -4308.87268835, -2454.75358719) km>),
'862': ('Saku',
 <EarthLocation (-3864.24552574, 3415.64676984, 3741.28760146) km>),
'863': ('Furukawa',
 <EarthLocation (-3775.40150468, 3498.50884254, 3750.344556) km>),
'864': ('Kumamoto',
 <EarthLocation (-3508.18294081, 4070.97254851, 3423.7839416) km>),
'865': ('Emmy Observatory, New Paltz',
 <EarthLocation (1304.74080321, -4586.64715655, 4221.62509893) km>),
'866': ('U.S. Naval Academy, Michelson',
 <EarthLocation (1159.83851697, -4827.36327205, 3990.8003209) km>),
'867': ('Saji Observatory',
 <EarthLocation (-3626.52032848, 3739.37841147, 3668.83196514) km>),
'868': ('Hidaka Observatory',
 <EarthLocation (-3755.17177602, 3737.4000931, 3539.35578404) km>),
'869': ('Tosa',
 <EarthLocation (-3660.3875998, 3866.7208687, 3499.6837719) km>),
'870': ('Campinas',
 <EarthLocation (4022.75767154, -4287.68274877, -2466.97409768) km>),
'871': ('Akou',
 <EarthLocation (-3670.22331192, 3748.89019452, 3615.00048886) km>),
'872': ('Tokushima',
 <EarthLocation (-3689.13964506, 3788.18487126, 3554.79087558) km>),
'873': ('Kurashiki Observatory',
 <EarthLocation (-3636.18438824, 3795.52897296, 3600.77724335) km>),
'874': ('Observatorio do Pico dos Dias, Itajuba',
 <EarthLocation (4126.27235742, -4211.05894345, -2429.98090308) km>),
'875': ('Yorii',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-3907.91177212, 3369.01962596, 3737.14181241) km>),
'876': ('Honjo',
 <EarthLocation (-3902.10765826, 3362.66269468, 3748.68624038) km>),
'877': ('Okutama',
 <EarthLocation (-3913.29352215, 3391.85795557, 3711.82060852) km>),
'878': ('Kagiya',
 <EarthLocation (-3820.57212112, 3573.45251855, 3636.74993603) km>),
'879': ('Tokai',
 <EarthLocation (-3845.55924453, 3541.93726214, 3641.85244563) km>),
'880': ('Rio de Janeiro',
 <EarthLocation (4283.75367748, -4025.93562902, -2466.04288968) km>),
'881': ('Toyota',
 <EarthLocation (-3835.004041, 3544.16006361, 3650.2078051) km>),
'882': ('JCPM Oi Station',
 <EarthLocation (-3839.69618074, 3536.25222853, 3653.46065497) km>),
'883': ('Shizuoka',
 <EarthLocation (-3911.67286977, 3470.32168045, 3639.68387905) km>),
'884': ('Kawane',
 <EarthLocation (-3885.36547666, 3488.68591853, 3650.8456188) km>),
'885': ('JCPM Yakiimo Station',
 <EarthLocation (-3917.41727588, 3469.89903185, 3633.94355575) km>),
'886': ('Mishima',
 <EarthLocation (-3935.50570001, 3428.72372097, 3653.3968736) km>),
'887': ('Ojima',
 <EarthLocation (-3906.56241515, 3355.82259959, 3750.21699326) km>),
'888': ('Gekko',
 <EarthLocation (-3941.36294773, 3426.75428428, 3649.37864729) km>),
'889': ('Karasuyama',
 <EarthLocation (-3932.67123375, 3283.2468065, 3786.82749964) km>),
'890': ('JCPM Tone Station',
 <EarthLocation (-3975.98363723, 3306.79068544, 3721.0051258) km>),
'891': ('JCPM Kimachi Station',
 <EarthLocation (-3888.75885804, 3164.44673867, 3929.50642433) km>),
'892': ('YGCO Hoshikawa and Nagano Stations',
 <EarthLocation (-3919.85636088, 3350.79717699, 3740.7773505) km>),
'893': ('Sendai Municipal Observatory',
 <EarthLocation (-3889.55505322, 3165.21650282, 3928.10323419) km>),
'894': ('Kiyosato',
 <EarthLocation (-3871.58435764, 3431.60587155, 3719.79327977) km>),
'895': ('Hatamae',
 <EarthLocation (-3879.21962273, 3172.80996457, 3932.63171146) km>),
'896': ('Yatsugatake South Base Observatory',
 <EarthLocation (-3867.70692082, 3437.79986414, 3717.94362004) km>),
'897': ('YGCO Chiyoda Station',
 <EarthLocation (-3918.21826389, 3347.31434721, 3745.56095325) km>),
'898': ('Fujieda',
 <EarthLocation (-3903.26812008, 3491.35899214, 3629.09617163) km>),
'899': ('Toma',
 <EarthLocation (-3657.87436292, 2801.71753483, 4395.1742067) km>),
'900': ('Moriyama',
 <EarthLocation (-3759.59788083, 3631.87738769, 3642.44561237) km>),
'901': ('Tajimi',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (-3814.79231463, 3546.45019613, 3669.02330925) km>),
'902': ('Ootake',
 <EarthLocation (-3547.77040023, 3909.79233168, 3566.78177314) km>),
'903': ('Fukuchiyama and Kannabe',
 <EarthLocation (-3697.76749058, 3675.00407858, 3662.19870266) km>),
'904': ('Go-Chome and Kobe-Suma',
 <EarthLocation (-3724.03487319, 3708.46825229, 3603.647405) km>),
'905': ('Nachi-Katsuura Observatory',
 <EarthLocation (-3820.09951491, 3698.75486545, 3510.5266048) km>),
'906': ('Cobram',
 <EarthLocation (-4273.03360641, 2918.47363285, -3722.9185669) km>),
'907': ('Melbourne',
 <EarthLocation (-4130.54643029, 2894.84050389, -3890.72735137) km>),
'908': ('Toyama',
 <EarthLocation (-3763.17358727, 3479.04385379, 3784.1486821) km>),
'909': ('Snohomish Hilltop Observatory',
 <EarthLocation (-2276.37203998, -3624.86942265, 4712.8054293) km>),
'910': ('Caussols-ODAS',
 <EarthLocation (4582.04112358, 556.6545381, 4388.85985107) km>),
'911': ('Collins Observatory, Corning Community College',
 <EarthLocation (1059.70821066, -4618.29792914, 4255.4930064) km>),
'912': ('Carbuncle Hill Observatory, Greene',
 <EarthLocation (1492.18826759, -4529.40262412, 4221.43375482) km>),
'913': ('Observatorio Kappa Crucis, Montevideo',
 <EarthLocation (2913.98859007, -4350.21935798, -3629.92532944) km>),
'914': ('Underwood Observatory, Hubbardston',
 <EarthLocation (1455.58440942, -4476.95658721, 4289.55225798) km>),
'915': ('River Oaks Observatory, New Braunfels',
 <EarthLocation (-782.62864846, -5484.5556678, 3150.35320841) km>),
'916': ('Oakley Observatory, Terre Haute',
 <EarthLocation (230.80044535, -4924.06467894, 4034.04408976) km>),
'917': ('Pacific Lutheran University Keck Observatory',
 <EarthLocation (-2331.79730296, -3667.55791257, 4652.72337876) km>),
'918': ('Badlands Observatory, Quinn',
 <EarthLocation (-965.97223914, -4494.14612966, 4407.9304807) km>),
'919': ('Desert Beaver Observatory',
 <EarthLocation (-1983.85532672, -4989.84680388, 3430.7998923) km>),
'920': ('RIT Observatory, Rochester',
 <EarthLocation (996.87430603, -4558.58306205, 4333.75274739) km>),
'921': ('SW Institute for Space Research, Cloudcroft',
 <EarthLocation (-1434.03859813, -5161.35508563, 3454.33521783) km>),
'922': ('Timberland Observatory, Decatur',
 <EarthLocation (259.58001204, -5245.01962125, 3608.04831953) km>),
'923': ('The Bradstreet Observatory, St. Davids',
 <EarthLocation (1234.88478122, -4730.63992002, 4082.39036822) km>),
'924': ('Observatoire du Cegep de Trois-Rivieres',
 <EarthLocation (1336.10067336, -4192.3916282, 4601.8258455) km>),
'925': ('Palominas Observatory',
 <EarthLocation (-1876.65684979, -5116.8296832, 3304.57656107) km>),
'926': ('Tenagra II Observatory, Nogales',
 <EarthLocation (-1941.13386681, -5088.89634555, 3310.38066574) km>),
'927': ('Madison-YRS',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (45.97753364, -4687.74987315, 4310.66389145) km>),
'928': ('Moonedge Observatory, Northport',
 <EarthLocation (1385.29951349, -4624.45134465, 4154.46331632) km>),
'929': ('Port Allen',
 <EarthLocation (-117.62427029, -5504.28743028, 3209.41475703) km>),
'930': ('S. S. Observatory, Pamatai',
 <EarthLocation (-5246.15171703, -3079.40561485, -1911.13221441) km>),
'931': ("Puna'auia",
 <EarthLocation (-5245.17116173, -3075.37966225, -1919.819237) km>),
'932': ('John J. McCarthy Obs., New Milford',
 <EarthLocation (1364.11782059, -4583.4557026, 4206.36221709) km>),
'933': ('Rockland Observatory, Sierra Vista',
 <EarthLocation (-1886.30860982, -5108.7243501, 3311.78385588) km>),
'934': ('Poway Valley',
 <EarthLocation (-2435.44571892, -4771.01741736, 3451.08236796) km>),
'935': ('Wyrick Observatory, Haymarket',
 <EarthLocation (1062.84364727, -4858.58678863, 3979.957488) km>),
'936': ('Ibis Observatory, Manhattan',
 <EarthLocation (-570.76167066, -4917.31339342, 4008.78666724) km>),
'937': ('Bradbury Observatory, Stockton-on-Tees',
 <EarthLocation (3702.49729067, -84.66796045, 5175.41170591) km>),
'938': ('Linhaceira',
 <EarthLocation (4874.01604416, -718.32398401, 4037.29693963) km>),
'939': ('Observatorio Rodeno',
 <EarthLocation (4909.89973781, -33.99531985, 4057.70697803) km>),
'940': ('Waterlooville',
 <EarthLocation (4030.25618287, -73.08552413, 4926.34545606) km>),
'941': ("Observatorio Pla D'Arguines",
 <EarthLocation (4910.28862262, -33.08954256, 4057.00538296) km>),
'942': ('Grantham',
 <EarthLocation (3852.98621912, -42.79793892, 5065.70774951) km>),
'943': ('Peverell',
 <EarthLocation (4063.81884549, -293.69365068, 4890.69167023) km>),
'944': ('Observatorio Geminis, Dos Hermanas',
 <EarthLocation (5054.11768551, -523.79452967, 3842.30453527) km>),
'945': ('Observatorio Monte Deva',
 <EarthLocation (4612.92361801, -452.41491422, 4367.36552938) km>),
'946': ('Ametlla de Mar',
 <EarthLocation (4825.36369408, 66.79794608, 4156.6318829) km>),
'947': ('Saint-Sulpice',
 <EarthLocation (4160.02129997, 154.31505415, 4816.19503007) km>),
'948': ('Pymoor',
 <EarthLocation (3893.69665858, 14.87604066, 5034.71000369) km>),
'949': ('Durtal',
 <EarthLocation (4302.28656337, -13.74885477, 4692.84186049) km>),
'950': ('La Palma',
 <EarthLocation (5327.27600253, -1718.85630587, 3051.74721039) km>),
'951': ('Highworth',
 <EarthLocation (3965.06907735, -117.79825265, 4977.88080302) km>),
'952': ('Marxuquera',
 <EarthLocation (4966.61109013, -20.95157848, 3988.2490661) km>),
'953': ('Montjoia',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4754.91812155, 177.1721029, 4234.63649841) km>),
'954': ('Teide Observatory',
 <EarthLocation (5390.41446417, -1597.6755066, 3006.78134454) km>),
'955': ('Sassoeiros',
 <EarthLocation (4918.37699777, -807.71584667, 3966.43583756) km>),
'956': ('Observatorio Pozuelo',
 <EarthLocation (4850.93081012, -322.9808289, 4115.8118061) km>),
'957': ('Merignac',
 <EarthLocation (4531.18393324, -51.35939996, 4473.43394769) km>),
'958': ('Observatoire de Dax',
 <EarthLocation (4618.33816735, -83.06375407, 4383.5213504) km>),
'959': ('Ramonville Saint Agne',
 <EarthLocation (4628.7582151, 118.40311012, 4372.08535076) km>),
'960': ('Rolvenden',
 <EarthLocation (4019.01842914, 42.84625281, 4935.84888019) km>),
'961': ('City Observatory, Calton Hill, Edinburgh',
 <EarthLocation (3573.39148137, -198.4948407, 5261.83546226) km>),
'962': ('Gandia',
 <EarthLocation (4965.0359183, -15.70216018, 3990.22628857) km>),
'963': ('Werrington',
 <EarthLocation (3880.41651174, -18.06266652, 5044.4685533) km>),
'964': ('Southend Bradfield',
 <EarthLocation (3983.6740256, -80.43422908, 4963.84890162) km>),
'965': ('Observacao Astronomica no Algarve, Portimao',
 <EarthLocation (5030.07719506, -760.6558134, 3834.40840166) km>),
'966': ('Church Stretton',
 <EarthLocation (3883.4271199, -189.64388449, 5039.3660437) km>),
'967': ('Greens Norton',
 <EarthLocation (3922.4401814, -69.98672226, 5012.25896145) km>),
'968': ('Haverhill',
 <EarthLocation (3927.54871227, 29.13371337, 5008.7509861) km>),
'969': ('London-Regents Park',
 <EarthLocation (3976.75394266, -10.73041779, 4969.8443504) km>),
'970': ('Chelmsford',
 <EarthLocation (3957.167179, 34.21594908, 4985.27944194) km>),
'971': ('Lisbon',
 <EarthLocation (4919.53560973, -795.69039218, 3967.45633948) km>),
'972': ('Dun Echt',
 <EarthLocation (3464.00779881, -146.19638326, 5335.69428872) km>),
'973': ('Harrow',
 <EarthLocation (3971.66265177, -23.0764183, 4973.86257671) km>),
'974': ('Genoa',
 <EarthLocation (4507.83567896, 707.68166565, 4441.54326269) km>),
'975': ('Observatorio Astronomico de Valencia',
 <EarthLocation (4929.68868443, -31.55104161, 4033.47005743) km>),
'976': ('Leamington Spa',
 <EarthLocation (3905.72055403, -104.53282401, 5024.56876586) km>),
'977': ('Markree',
 <EarthLocation (3700.78382347, -549.89682931, 5148.24084229) km>),
'978': ('Conder Brow',
 <EarthLocation (3750.37515745, -180.44457612, 5138.6927712) km>),
'979': ('South Wonston',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4010.51180957, -93.13326062, 4942.09945445) km>),
'980': ('Lancaster',
 <EarthLocation (3750.00808522, -182.09388471, 5138.8649809) km>),
'981': ('Armagh',
 <EarthLocation (3700.35839054, -431.27479274, 5159.78527026) km>),
'982': ('Dunsink Observatory, Dublin',
 <EarthLocation (3788.97620929, -420.84393283, 5096.38658848) km>),
'983': ('San Fernando',
 <EarthLocation (5105.38191368, -555.05021221, 3769.90630218) km>),
'984': ('Eastfield',
 <EarthLocation (4024.3108508, -191.91114415, 4928.11857815) km>),
'985': ('Telford',
 <EarthLocation (3874.37310334, -167.01117631, 5047.0198081) km>),
'986': ('Ascot',
 <EarthLocation (3979.01036589, -86.82231504, 4968.568723) km>),
'987': ('Isle of Man Observatory, Foxdale',
 <EarthLocation (3729.09728497, -301.77202664, 5148.49596777) km>),
'988': ('Glasgow',
 <EarthLocation (3576.04125202, -268.50729604, 5256.92429677) km>),
'989': ('Wilfred Hall Observatory, Preston',
 <EarthLocation (3823.77238206, -154.24701902, 5083.375189) km>),
'990': ('Madrid',
 <EarthLocation (4853.89509078, -312.85797353, 4113.06920719) km>),
'991': ('Liverpool (since 1867)',
 <EarthLocation (3805.45974822, -204.24455083, 5097.34330903) km>),
'992': ('Liverpool (before 1867)',
 <EarthLocation (3804.4384817, -199.41546344, 5098.0449041) km>),
'993': ('Woolston Observatory',
 <EarthLocation (4014.22895341, -175.57668832, 4936.87576025) km>),
'994': ('Godalming',
 <EarthLocation (4006.96339019, -42.81565528, 4945.6074298) km>),
'995': ('Durham',
 <EarthLocation (3686.36885754, -101.82992187, 5186.63722703) km>),
'996': ('Oxford',
 <EarthLocation (3955.09548165, -86.41789641, 4986.36372523) km>),
'997': ('Hartwell',
 <EarthLocation (3947.63235368, -58.56860167, 4994.081271) km>),
'998': ('London-Mill Hill',
 <EarthLocation (3968.78572209, -16.79560392, 4976.26075622) km>),
'999': ('Bordeaux-Floirac',
 <EarthLocation (4530.39004665, -41.71072098, 4474.45444961) km>),
'A00': ('Gravesend',
 <EarthLocation (3984.65483144, 26.21897396, 4963.52999477) km>),
'A01': ('Masia Cal Maciarol Modul 2',
 <EarthLocation (4745.82661866, 61.63749174, 4247.58411652) km>),
'A02': ('Masia Cal Maciarol Modul 8',
 <EarthLocation (4745.82661866, 61.63749174, 4247.58411652) km>),
'A03': ('Torredembarra',
 <EarthLocation (4808.31735125, 117.5127449, 4175.1284802) km>),
'A04': ('Saint-Caprais',
 <EarthLocation (4603.32719295, 138.07906845, 4398.10814972) km>),
'A05': ('Belesta',
```

(continues on next page)

(continued from previous page)

```
 <EarthLocation (4635.91159889, 147.10676195, 4363.72999129) km>),
'A06': ('Mataro',
 <EarthLocation (4774.28923099, 203.58295547, 4210.33579644) km>),
'A07': ('Gretz-Armainvilliers',
 <EarthLocation (4209.20191691, 201.77011853, 4771.80319655) km>),
'A08': ('Malibert',
 <EarthLocation (4633.25937831, 233.47039234, 4363.02839622) km>),
'A09': ('Quincampoix',
 <EarthLocation (4147.26882722, 85.44632748, 4828.95130407) km>),
'A10': ('Observatorio Astronomico de Corbera',
 <EarthLocation (4798.61562458, 161.54252479, 4184.88702981) km>),
'A11': ('Wormhout',
 <EarthLocation (4028.63391858, 173.90739371, 4925.13361003) km>),
'A12': ('Stazione Astronomica di Sozzago',
 <EarthLocation (4434.21970793, 682.30788832, 4518.48910746) km>),
'A13': ('Observatoire Naef, Marly',
 <EarthLocation (4343.50339001, 544.04539814, 4624.21310637) km>),
'A14': ('Les Engarouines Observatory',
 <EarthLocation (4575.23596509, 415.28412008, 4410.03526591) km>),
'A15': ('Josef Bresser Sternwarte, Borken',
 <EarthLocation (3920.50699834, 467.29784096, 4992.48673675) km>),
'A16': ('Tentlingen',
 <EarthLocation (4342.3673943, 547.96799884, 4624.85092007) km>),
'A17': ('Guidestar Observatory, Weinheim',
 <EarthLocation (4099.07731268, 625.89397667, 4830.27157843) km>),
'A18': ('Herne',
 <EarthLocation (3945.11172363, 496.711698, 4970.35460136) km>),
'A19': ('Koln',
 <EarthLocation (3998.01632456, 496.16531387, 4928.19511579) km>),
'A20': ('Sogel',
 <EarthLocation (3827.32704302, 505.15957329, 5060.13963591) km>),
'A21': ('Irmtraut',
 <EarthLocation (4020.67421433, 569.22714372, 4902.36366094) km>),
'A22': ('Starkenburg Sternwarte-SOHAS',
 <EarthLocation (4090.84326326, 622.56040539, 4837.68935176) km>),
'A23': ('Weinheim',
 <EarthLocation (4100.14242927, 625.0446177, 4829.46155503) km>),
'A24': ('New Millennium Observatory, Mozzate',
 <EarthLocation (4410.04390794, 694.38798018, 4540.27682345) km>),
'A25': ('Nova Milanese',
 <EarthLocation (4413.90470931, 714.3036444, 4533.38843549) km>),
'A26': ('Darmstadt',
 <EarthLocation (4077.09518704, 620.77833012, 4849.31669551) km>),
...}
```

This Jupyter-Notebook was designed as a tutorial for how to work with the `Observer` Class. More information about the other classes, please refer to their specif Jupyter-Notebook. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The End

9.5 LightCurve Class

The `LightCurve` Class within SORA was created to reduce and analyze stellar occultations' light curves. Here we present some useful tasks that allows the user: - set the model's parameters - prepare the light curve to be reduced - fit one or all parameters simultaneously.

The documentation here contains the details about every step.

This Jupyter-Notebook was designed as a tutorial for how to work with the `LightCurve` Class. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The `LightCurve` Docstring was designed to help the users. Also, each function has its Docstring containing its main purpose and the needed parameters to set (physical description and formats). Please, do not hesitate to use it.

9.5.1 0. Index

[1]:
```
## SORA package
from sora import LightCurve

## Other main and necessary packages
from astropy.time import Time
import astropy.units as u
import numpy as np
import matplotlib.pylab as pl
import os
```

```
SORA version: 0.3
```

9.5.2 1. Instantiating a LightCurve Object

The `LightCurve` Class can be instantiated in different ways.

The LightCurve main goal is to reduce and analyse stellar occultations light curves, which is a file composed of two or three columns: time, flux, and flux uncertainty. An ASCII file can be used as input, where the first column is the time (Julian Date or seconds relative to a reference time), the second column is the flux for each time, and the third is the flux uncertainty (optional). Another option is to furnish theses parameters directly as NumPy arrays. If the time provided is in JD the reference time will be set as 00:00:00.000 of the occultation date, otherwise the user should add the user reference time.

Other observational parameters needed are the exposure time for the respective light curve (in seconds) and the filter parameters (central wavelength and bandwidth, in microns). If the filter parameter is not furnished the default values are used: λ_0 as 0.7 and $\Delta\lambda$ as 0.3 microns.

If the parameters of the light curve were already calculated using SORA or other method, the user could instantiate the `LightCurve` by only providing the instants of immersion and emersion and their uncertainties, with no need to provide a light curve file. This is useful for example in cases where the user does not want to redo the fitting process and go directly to the chords and ellipse fit procedure within the `Occultation` Class.

The user can also instantiate a `LightCurve` Object for the negative observations. In this case, it is just needed the initial and the end time of the observational sequence. In this case, an Error (and a warning message) will be raised for all functions in this module that require times and fluxes, but the object will be instantiated and can be used for other purposes.

[2]:
```
LightCurve?
```

```
Init signature: LightCurve(name='', **kwargs)
Docstring:
Defines a Light Curve.

Parameters
----------
name : `str`
    The name of the LightCurve. Each time an LightCurve object is defined
    the name must be different.

tref : `astropy.time.Time`, `str`, `float`
    Instant of reference.

    Format: `Julian Date`, string in ISO format or Time object.
    Required only if LightCurve have input fluxes and given time is
    not in Julian Date.

central_bandpass : `int`, `float`, otpional, default=0.7
    The center band pass of the detector used in observation. Value in microns.

delta_bandpass : `int`, `float`, optional, default=0.3
    The band pass width of the detector used in observation. Value in microns.

exptime : `int`, `float`
    The exposure time of the observation, in seconds.
    *NOT* required in cases *2*, *3* and *4* below.
    *Required* in case *1* below.

**kwargs: `int`, `float`
```

(continues on next page)

(continued from previous page)

```
    Object velocity, distance, and star diameter.

    Note
    ----
    vel : `int`, `float`
        Velocity in km/s.

    dist : `int`, `float`
        Object distance in AU.

    d_star : `float`
        Star diameter, in km.


Warning
-------
Input data must be one of the 4 options below:

1) Input data from file with time and flux
    `file (str)`: a file with the time and flux. A third column with the error in
    flux can also be given.

    `usecols (int, tuple, array)`: Which columns to read, with the first being the
    time, the seconds the flux and third the flux error (optional).

    **Example:**

    >>> LightCurve(name, file, exptime) # dflux can also be given


2) Input data when file is not given:
    `time`: time must be a list of times, in seconds from tref, or Julian Date, or
    a Time object.

    `flux`: flux must be a list of fluxes. It must have the same lenght as time.

    `dflux`: if file not given, dflux must be a list of fluxes errors. It must
    have the same lenght as time. (not required)

    **Example:**

    >>> LightCurve(name, flux, time, exptime) # dflux can also be given


Cases for when `time` and `flux` are not given.


3) Input for a positive occultation:
    `immersion`: The instant of immersion.

    `emersion`: The instant of emersion.
```

(continues on next page)

(continued from previous page)

```
    `immersion_err`: Immersion time uncertainty, in seconds.

    `emersion_err`: Emersion time uncertainty, in seconds.

    **Example:**

    >>> LightCurve(name, immersion, immersion_err, emersion, emersion_err)


4) Input for a negative occultation:
    `initial_time`: The initial time of observation.

    `end_time`: The end time of observation.

    **Example:**

    >>> LightCurve(name, initial_time, end_time)
File:           ~/Documentos/códigos/SORA/sora/lightcurve/core.py
Type:           type
Subclasses:
```

Example of `LightCurve` intantiate with an ASCII file

[3]:
```
# The file lc_example.dat contains time in JD and flux
lc_1a = LightCurve(name='Example 1a', file='input/lightcurves/lc_example.dat', exptime=0.
↪100)
```

[4]:
```
# The file lc_example_trend.dat contains time in seconds and flux, so it is needed a␣
↪reference time
lc_1b = LightCurve(name='Example 1b', file='input/lightcurves/lc_example_trend.dat',
                   tref='2017-06-22 00:00', exptime=0.100)
```

[5]:
```
# If the file has more columns, the user can set which columns to use. Always in a sense,
↪ time, flux, flux error [optional].
lc_1c = LightCurve(name='Example 1c', file='input/lightcurves/lc_example_columns.dat',
                   usecols=[2,4], exptime=0.100)
```

Example of `LightCurve` intantiate with NumPy arrays with times as Julian Dates

[6]:
```
time, flux = np.loadtxt('input/lightcurves/lc_example_2.dat', unpack=True)

print('first time: ', time[0])

lc_2a = LightCurve(name='Example 2a', time=time, flux=flux, exptime=0.075)
```

```
first time:  2457927.38872249
```

Example of `LightCurve` instantiate with `NumPy` arrays with times as seconds relative to a reference time

[7]:
```
time, flux = np.loadtxt('input/lightcurves/lc_example_2.dat',unpack=True)
```

(continues on next page)

(continued from previous page)

```
tref = Time('2017-06-22 21:00').jd

time_sec = (time - tref)*u.d.to(u.s)

print('first time: ',time_sec[0])

lc_2b = LightCurve(name='Example 2b',time=time_sec,flux=flux,tref='2017-06-22 21:00',
↪exptime=0.075)
```

```
first time:  1185.6231406331062
```

Example of `LightCurve` intantiate with the required parameters (positive)

[8]:
```
lc_3 = LightCurve(name='Example 3',
                  initial_time='2017-06-22 21:16:00.094',
                  end_time ='2017-06-22 21:28:00.018',
                  immersion='2017-06-22 21:21:15.628',immersion_err=0.010,
                  emersion ='2017-06-22 21:21:19.988',emersion_err=0.040)
```

Example of `LightCurve` intantiate with the required parameters (negative)

[9]:
```
lc_4 = LightCurve(name='Example 4',
                  initial_time='2017-06-22 21:16:00.094',
                  end_time ='2017-06-22 21:28:00.018')
```

After the `LightCurve` was instantiated, some parameters can be easily retrieved

[10]:
```
print('Name: {}'.format(lc_1a.name))
print('Initial time: {} UTC'.format(lc_1a.initial_time.iso))
print('Mean time:    {} UTC'.format(lc_1a.time_mean.iso))
print('End time:     {} UTC'.format(lc_1a.end_time.iso))
```

```
Name: Example 1a
Initial time: 2017-06-22 21:20:00.056 UTC
Mean time:    2017-06-22 21:21:40.007 UTC
End time:     2017-06-22 21:23:19.958 UTC
```

For cases where the immersion and emersion time were instanciated (or calculated), the mean time is between these values.

[11]:
```
print('Initial time:   {} UTC'.format(lc_3.initial_time.iso))
print('Immersion time: {} UTC'.format(lc_3.immersion.iso))
print('Mean time:      {} UTC'.format(lc_3.time_mean.iso))
print('Emersion time:  {} UTC'.format(lc_3.emersion.iso))
print('End time:       {} UTC'.format(lc_3.end_time.iso))
```

```
Initial time:   2017-06-22 21:16:00.094 UTC
Immersion time: 2017-06-22 21:21:15.628 UTC
Mean time:      2017-06-22 21:21:17.808 UTC
Emersion time:  2017-06-22 21:21:19.988 UTC
End time:       2017-06-22 21:28:00.018 UTC
```

If the `LightCurve` were instantiated with times and fluxes, the exposure time and the mean cycle can also be obtained

[12]:
```
print('Exposure time: {:.4f} s'.format(lc_1a.exptime))
print('Cycle time:    {:.4f} s'.format(lc_1a.cycle))
```

```
Exposure time: 0.1000 s
Cycle time:    0.1002 s
```

9.5.3 2. Set and/or modifing model parameters

In order to create a stellar occultation light curve model, some physical parameters are needed. First, we need to 'project' the time axis of the light curve in the sky plane using the event's velocity (vel in km/s). Then the Fresnel diffraction should be applied. That diffraction works in a scale that depends on the object distance ($dist$ in AU) and the observational wavelength (λ_0 and $\Delta\lambda$, in microns). The last parameter needed is the occulted star apparent diameter at the object distance (d_star in km). In all cases, the user can provide each parameter with its own unit, otherwise SORA will consider the standard units described above.

Let's start with the shadow velocity during the occultation, in km/s

[13]:
```
lc_1a.set_vel(vel=22.0)
print('Vel: {:3.1f} km/s'.format(lc_1a.vel))

lc_1b.set_vel(vel=22.0*u.km/u.s)
print('Vel: {:3.1f} km/s'.format(lc_1b.vel))

lc_2a.set_vel(vel=22000.0*u.m/u.s)
print('Vel: {:3.1f} km/s'.format(lc_2a.vel))
```

```
Vel: 22.0 km/s
Vel: 22.0 km/s
Vel: 22.0 km/s
```

Now, the object distance at occultation time, in AU

[14]:
```
lc_1a.set_dist(dist=15)
print('Distance: {:3.1f} AU'.format(lc_1a.dist))

lc_1b.set_dist(dist=15*u.AU)
print('Distance: {:3.1f} AU'.format(lc_1b.dist))

lc_2a.set_dist(dist=2243968060.5*u.km)
print('Distance: {:3.1f} AU'.format(lc_2a.dist))
```

```
Distance: 15.0 AU
Distance: 15.0 AU
Distance: 15.0 AU
```

And then, the observational wavelength, central value (:math:`lambda_0`) and its bandwidth (:math:`Deltalambda`), in microns

Default value set as λ_0 = 0.7 and $\Delta\lambda$ = 0.3 microns, no filter (Clear).

[15]:
```
print('Observational wavelength centred at {:1.3f} with a bandwidth of {:1.3f} microns'
      .format(lc_1a.central_bandpass,lc_1a.delta_bandpass))
```

(continues on next page)

(continued from previous page)

```
lc_1b.set_filter(central_bandpass=0.8, delta_bandpass=0.2)
print('Observational wavelength centred at {:1.3f} with a bandwidth of {:1.3f} microns'
      .format(lc_1b.central_bandpass,lc_1b.delta_bandpass))

lc_2a.set_filter(central_bandpass=0.8*u.micrometer, delta_bandpass=0.2*u.micrometer)
print('Observational wavelength centred at {:1.3f} with a bandwidth of {:1.3f} microns'
      .format(lc_2a.central_bandpass,lc_2a.delta_bandpass))

lc_4.set_filter(central_bandpass=800*u.nm, delta_bandpass=200*u.nm)
print('Observational wavelength centred at {:1.3f} with a bandwidth of {:1.3f} microns'
      .format(lc_4.central_bandpass,lc_4.delta_bandpass))
```

```
Observational wavelength centred at 0.700 with a bandwidth of 0.300 microns
Observational wavelength centred at 0.800 with a bandwidth of 0.200 microns
Observational wavelength centred at 0.800 with a bandwidth of 0.200 microns
Observational wavelength centred at 0.800 with a bandwidth of 0.200 microns
```

With the model parameters, the mean Fresnel Scale (:math:`F_S`) is automatically calculated.

$F_S = \frac{1}{2} \cdot \left(\sqrt{\frac{(\lambda+0.5.\Delta\lambda).dist}{2}} + \sqrt{\frac{(\lambda-0.5.\Delta\lambda).dist}{2}} \right)$

[16]:
```
print('Fresnel Scale: {:1.3f} km'.format(lc_1a.fresnel_scale))
```

```
Fresnel Scale: 0.881 km
```

Last, let's set the stellar diameter projected at the body distance, in km

[17]:
```
lc_1a.set_star_diam(d_star=0.2)
print('Stellar diameter: {:1.2f} km'.format(lc_1a.d_star))

lc_1b.set_star_diam(d_star=0.2*u.km)
print('Stellar diameter: {:1.2f} km'.format(lc_1b.d_star))

lc_2a.set_star_diam(d_star=200*u.m)
print('Stellar diameter: {:1.2f} km'.format(lc_2a.d_star))
```

```
Stellar diameter: 0.20 km
Stellar diameter: 0.20 km
Stellar diameter: 0.20 km
```

If the user instantiated the `LightCurve` object without times and flux and want to do it later (or redo it)

[18]:
```
lc_4.set_flux(file='input/lightcurves/lc_example_3.dat',exptime=0.30, tref='2018-05-20␣
↪00:00:00.000')
```

9.5.4 3. Preparing the LightCurve

With the `LightCurve` object instantiated, the user may want to improve the quality of his curve using some processes.

For SORA v0.2 the only process implemented is the normalization of a light curve using a polinomial fit.

Other processes will be developed for future use, such as the binning of the light curve (sum, mean and median), a Savitzky–Golay filter, among others.

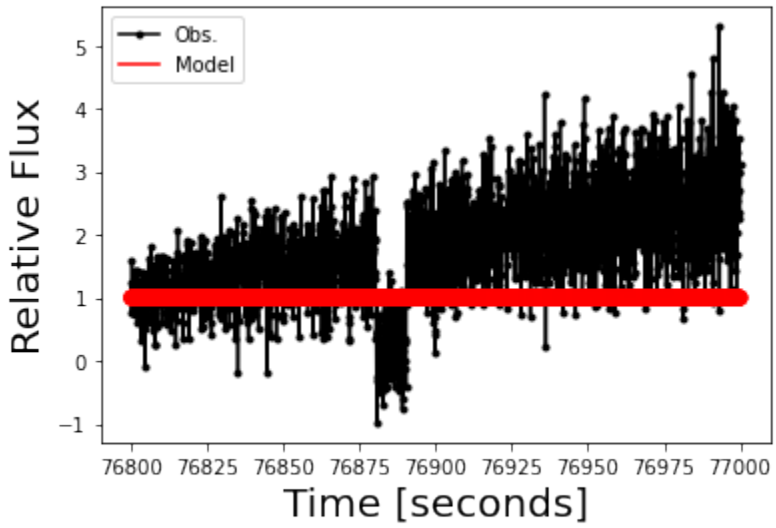
First, let's check the expected magnitude drop and bottom flux

[19]:
```
lc_1b.calc_magnitude_drop(mag_obj=18.6,mag_star=14.3)

print('Expected magnitude drop: {:.3f}'.format(lc_1b.mag_drop))
print('Expected bottom flux:    {:.3f}'.format(lc_1b.bottom_flux))
```

```
Expected magnitude drop: 4.320
Expected bottom flux:    0.019
```

Now, let's look at this light curve

[20]:
```
lc_1b.plot_lc()
```



There is a linear trend at this light curve.

3.1. Light curve normalization using a polinomial fit

The user can remove this linear trend using the function `LightCurve.normalize()`. That function can be used automatically or with the desired inputed parameters.

[21]:
```
lc_1b.normalize?
```

```
Signature:
lc_1b.normalize(
    poly_deg=None,
    mask=None,
    flux_min=0.0,
    flux_max=1.0,
    plot=False,
```

(continues on next page)

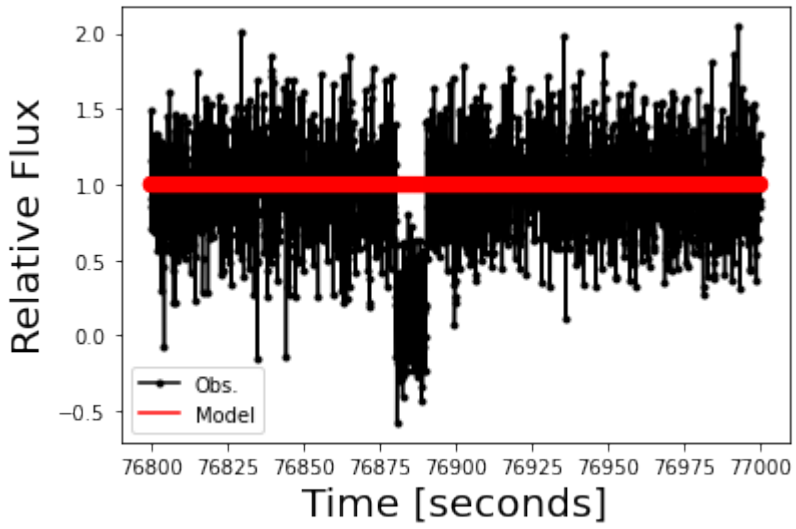
(continued from previous page)

```
)
Docstring:
Returns the normalized flux within the flux min and flux max defined scale.

Parameters
----------
poly_deg : `int`
    Degree of the polynomial to be fitted.

mask : `bool` array
    Which values to be fitted.

flux_min : `int`, `float`
    Event flux to be set as 0.

flux_max : `int`, `float`
    Baseline flux to be set as 1.

plot : `bool`
    If True plot the steps for visual aid.
File:      ~/Documentos/códigos/SORA/sora/lightcurve/core.py
Type:      method
```

Let's try the automatic mode

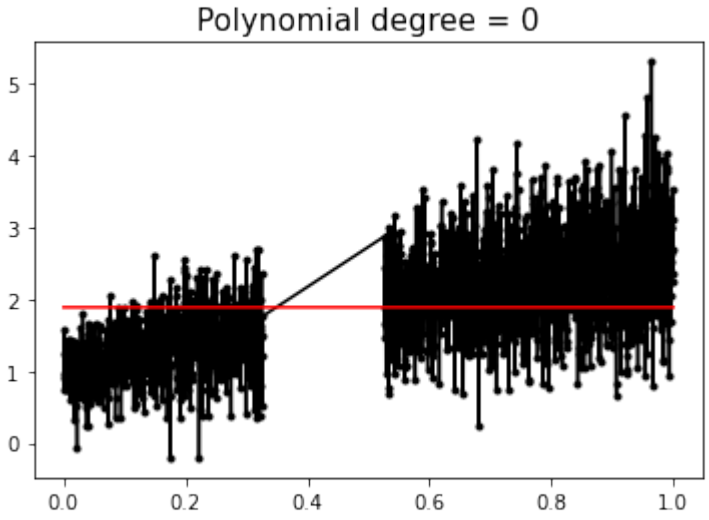
[22]:
```
lc_1b.normalize()
lc_1b.plot_lc()
```

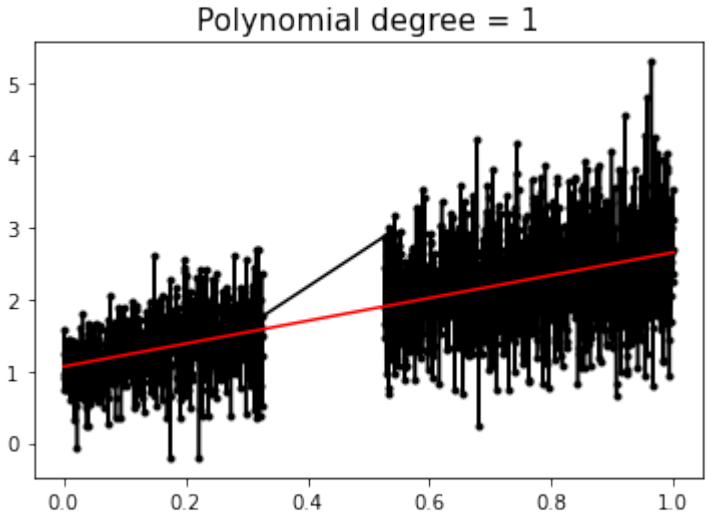
```
Normalization using a 1 degree polynomial
There is no improvement with a 2 degree polynomial
```



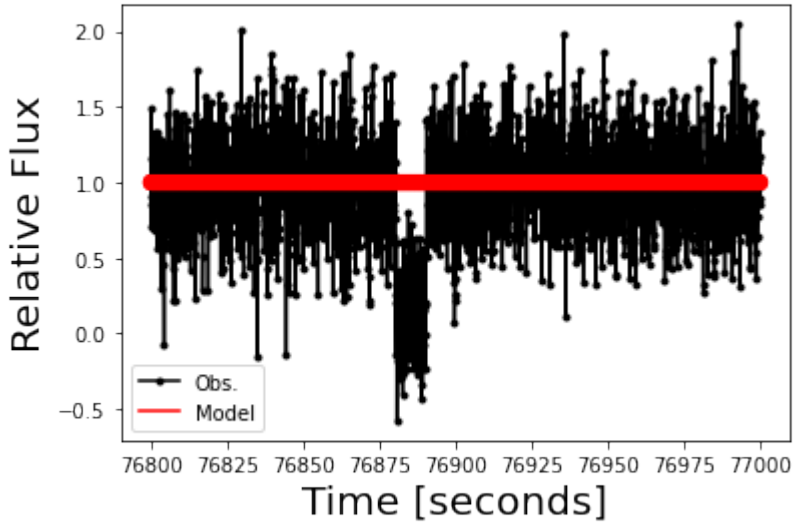
If the user wants to see the steps inside the automatic mode, just set the `plot` parameter as True

[23]:
```
lc_1b.normalize(plot=True)
lc_1b.plot_lc()
```





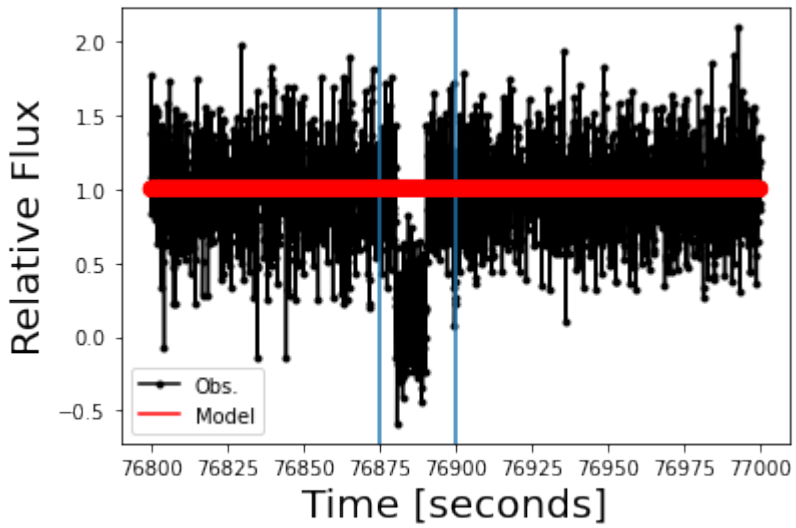

```
Normalization using a 1 degree polynomial
There is no improvement with a 2 degree polynomial
```



As an standard procedure within the SORA library the user can control all the parameters

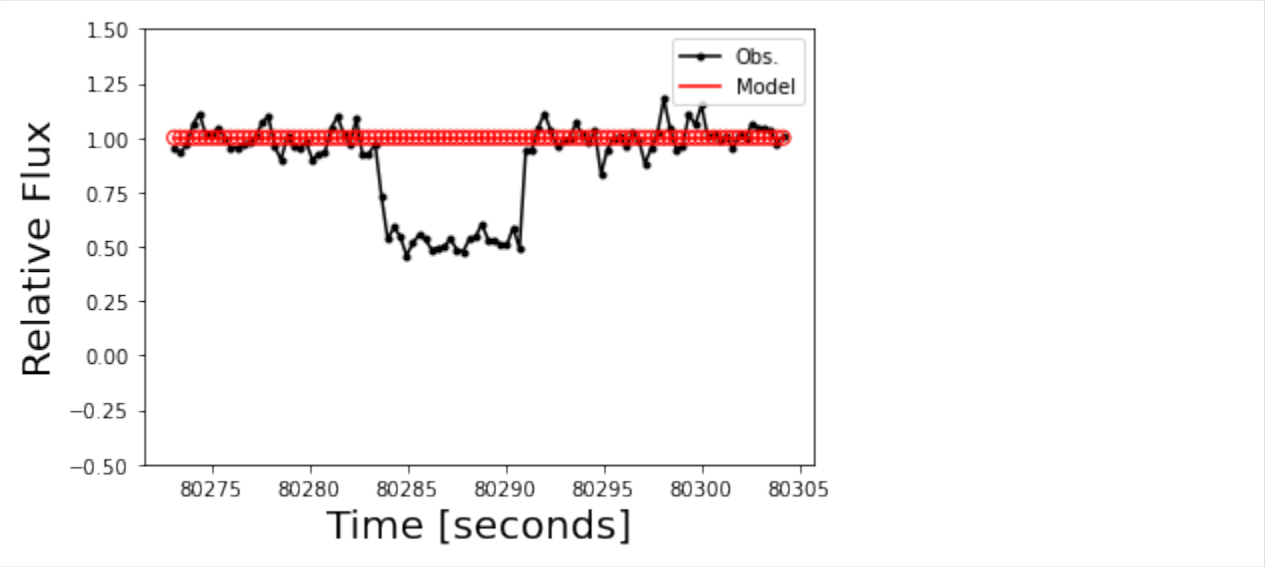
This is useful when the automatic mode does not work correctly (particular cases), or for better control of the procedure.

[24]:
```
mask = (lc_1b.time < 76875) + (lc_1b.time > 76900)
lc_1b.normalize(poly_deg=5,mask=mask)

lc_1b.plot_lc()
pl.axvline(76875)
pl.axvline(76900)
pl.show()
```



Also, the user can control the maximum and/or minimum value of the flux

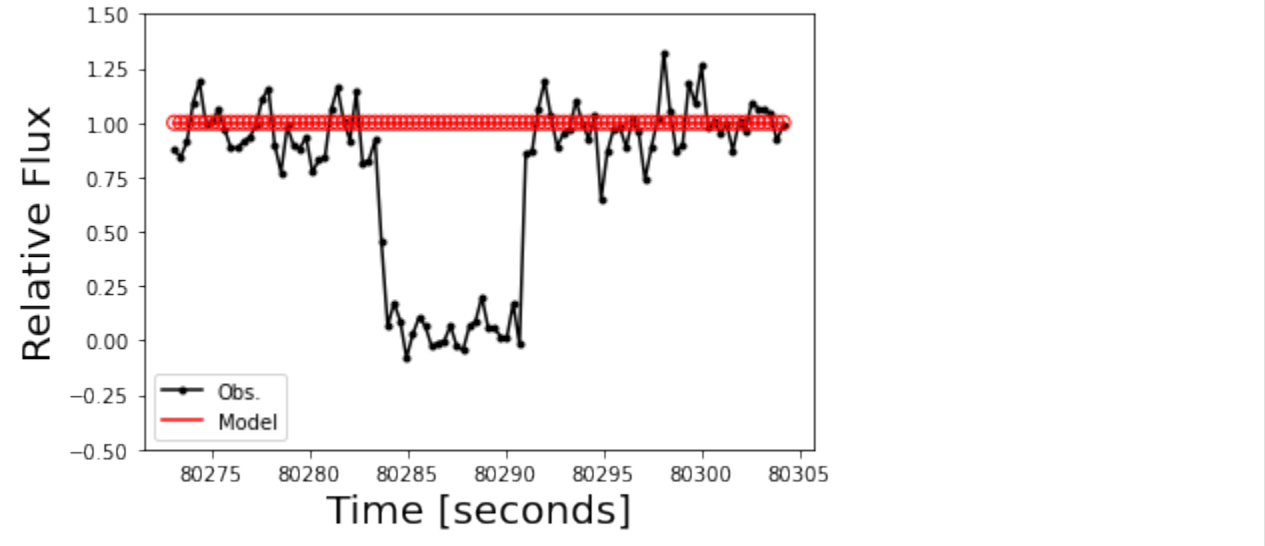
[25]:
```
lc_4.plot_lc()
pl.ylim(-0.5,1.5)
pl.show()
```



In this case the bottom flux goes to 0.5, instead of the expected 0.0. This can be solved setting the `flux_min` parameter

[26]:
```
lc_4.normalize(flux_min=0.50)

lc_4.plot_lc()
pl.ylim(-0.5,1.5)
pl.show()
```

```
Normalization using a 0 degree polynomial
There is no improvement with a 1 degree polynomial
```



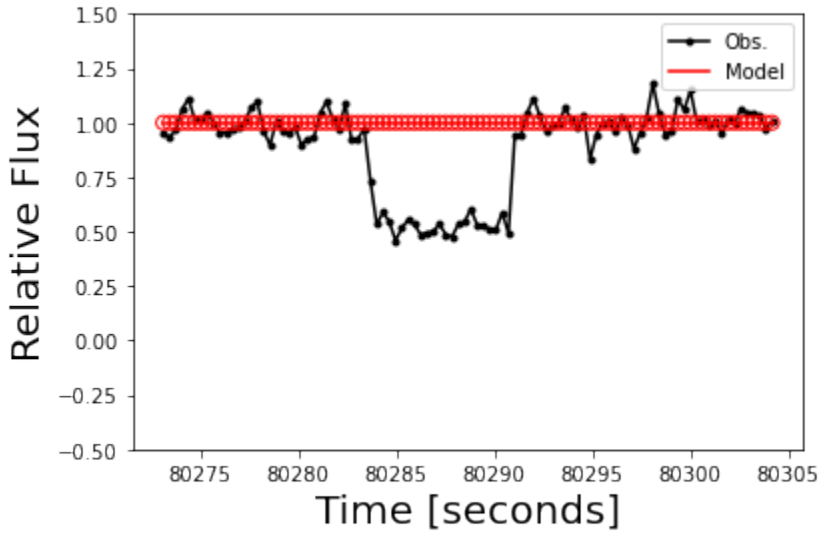
If the user is not satisfied with the normalization, it is possible to throw away and redo the normalization process and to return to the initial stage using the `LightCurve.reset_flux()`

[27]:
```
lc_4.reset_flux()
```

(continues on next page)

(continued from previous page)

```
lc_4.plot_lc()
pl.ylim(-0.5,1.5)
pl.show()
```



9.5.5 4. Automatic detection of Occultations

Before considering any effect (Fresnell scale, star diameter, etc), the user can choose that SORA detect the events in the light curve automatically. This process is done using the BLS algorithm (Kovacs et al., 2002). This function will always find events and give their Signal to Noise Raito (SNR). The user will consider the values for the events found and determine the significance of each event. This is a useful tool for detection of low depth events or to check for secondary events, for example.

The result of the `LightCurve.occ_detect()` will be a python dictionary containing: a rank for the detection; duration; central, immersion and emersion times; time uncertainty; flux depth and its uncertainty; baseline flux and its uncertainty; SNR; and a boollean mask with the values inside the detection. All the times are in seconds. The automatic mode will determine only one detection (the most significant).

[28]:
```
lc_2a.occ_detect?
```

```
Signature:
lc_2a.occ_detect(
    maximum_duration=None,
    dur_step=None,
    snr_limit=None,
    n_detections=None,
    plot=False,
)
Docstring:
Detects automatically the occultation event in the light curve.

Detects a 'square well' shaped transit. All parameters are optional.

Parameters
----------
```

(continues on next page)

(continued from previous page)

```
maximum_duration : `float`, default: light curve time span
    Maximum duration of the occultation event.

dur_step : `float`, default: 1/2 of sampling rate
    Step size to sweep occultation duration event.

snr_limit : `float`, default=None
    Minimum occultation SNR.

n_detections : `int`, default=1
    Number of detections regardless the SNR. `n_detections` is
    superseded by `snr_limit`.

plot : `bool`
    True if output plots are desired.


Returns
-------
OrderedDict : `dict`
    An ordered dictionary of :attr:`name`::attr:`value` pairs for each
    parameter.

Examples
--------
>>> lc = LightCurve(time=time, flux=flux, exptime=0.0, name='lc_example')
>>> params = lc.occ_detect()
>>> params
{'rank': 1,
'occultation_duration': 40.1384063065052,
'central_time': 7916.773870512843,
'immersion_time': 7896.7046673595905,
'emersion_time': 7936.843073666096,
'time_err': 0.05011036992073059,
'depth': 0.8663887801707082,
'depth_err': 0.10986223384336465,
'baseline': 0.9110181732552853,
'baseline_err': 0.19045768512595365,
'snr': 7.886138392251848,
'occ_mask': array([False, False, False, ..., False, False, False])}
File:      ~/Documentos/códigos/SORA/sora/lightcurve/core.py
Type:      method
```

For visual aid the user can set the `plot` parameter as True

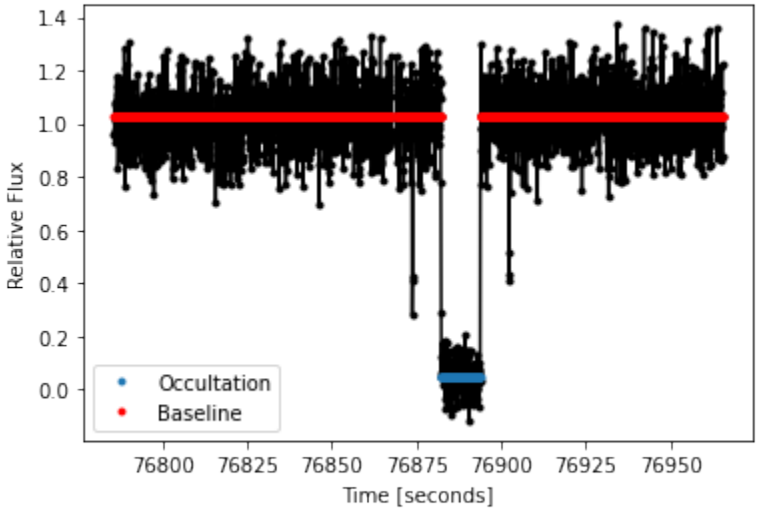
[29]:
```
preliminary = lc_2a.occ_detect(plot=True)
preliminary
```

[29]:
```
{'rank': 1,
 'occultation_duration': 11.52958944439888,
 'central_time': 76888.023570925,
 'immersion_time': 76882.2587762028,
```

(continues on next page)

(continued from previous page)

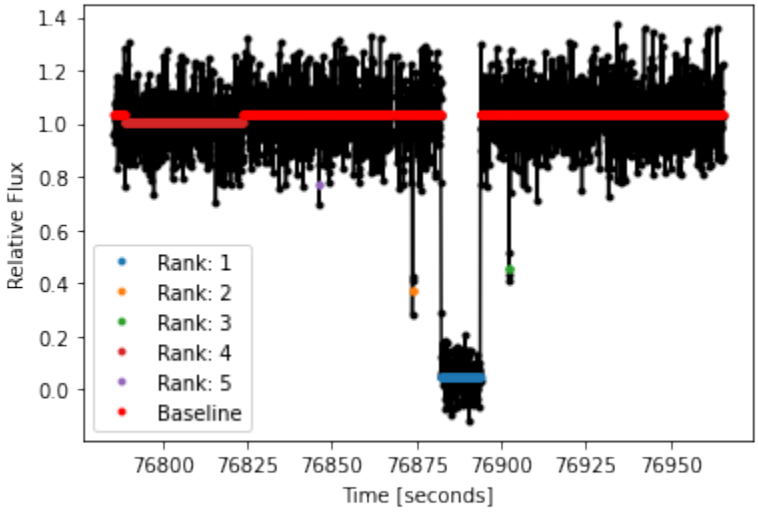
```
'emersion_time': 76893.7883656472,
'time_err': 0.04017278552055359,
'depth': 0.9797601789722584,
'depth_err': 0.06536566413075075,
'baseline': 1.0271920160757066,
'baseline_err': 0.11532581974159437,
'snr': 14.988911869883966,
'occ_mask': array([False, False, False, ..., False, False, False])}
```



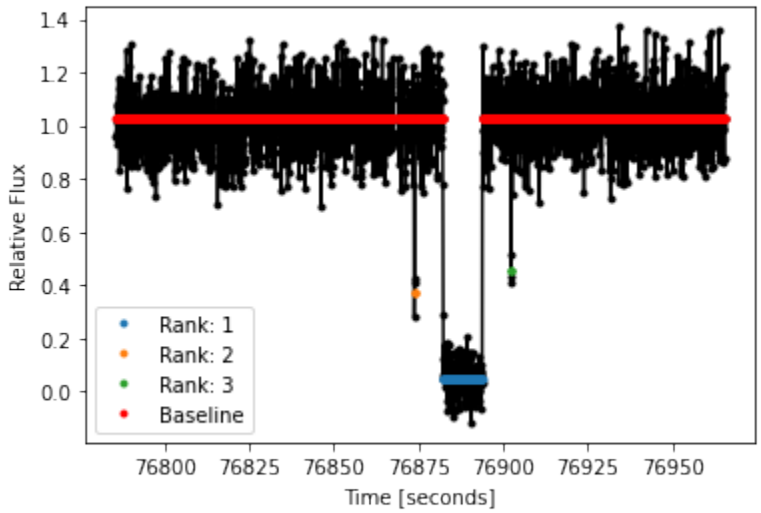
The user can choose to detect more than one event, two parameters can be used to control this `n_detections` and `snr_limit`.

With the `n_detections`, the user will set the number of detections they want. On the other hand `snr_limit` will obtain all the detection within the set limit. In all cases is up to the user to confirm if these detections have physical meaning.

[30]:
```
preliminary = lc_2a.occ_detect(n_detections=5, plot=True)
```



[31]:
```
preliminary = lc_2a.occ_detect(snr_limit=5, plot=True)
```

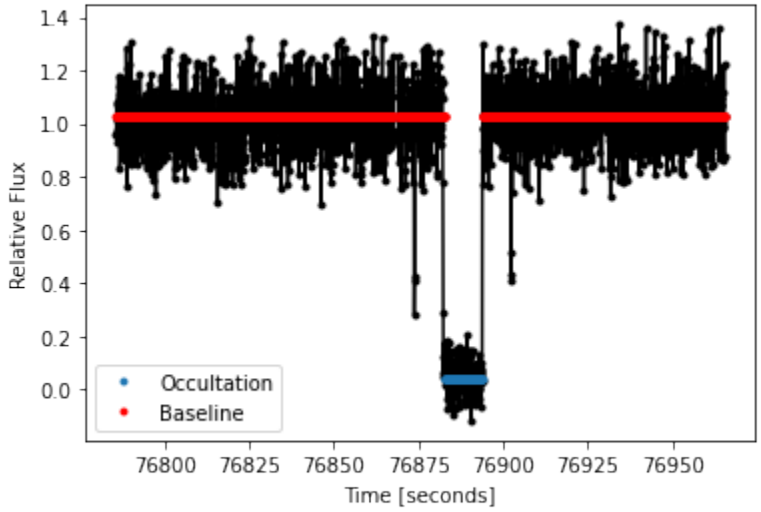


Other two parameters can be used for otimization purpose

The first is the `maximum_duration`, and this will set an upper limit to the size of the occultation (in seconds). The user can also set the `dur_step`, which is the step size in seconds of the occultation search. Those parameters will decrease the CPU time of these searches.

[32]:
```
preliminary = lc_2a.occ_detect(maximum_duration=12,dur_step=1, plot=True)
preliminary
```

[32]:
```
{'rank': 1,
 'occultation_duration': 11.0,
 'central_time': 76888.3231406331,
 'immersion_time': 76882.8231406331,
 'emersion_time': 76893.8231406331,
 'time_err': 0.04017278552055359,
 'depth': 0.9802503938156636,
 'depth_err': 0.062074094327636595,
 'baseline': 1.0241496004823303,
 'baseline_err': 0.1266541830883394,
 'snr': 15.791618136895426,
 'occ_mask': array([False, False, False, ..., False, False, False])}
```



9.5.6 5. Complete occultation light curve model and fitting

The model considers a sharp-edge occultation model (geometric) convolved with Fresnel diffraction, stellar diameter (projected at the body distance), CCD bandwidth, and finite integration time. It is created using a smaller time resolution than the observational light curve. By default, the model will have a resolution that is 1/10 of the mean Fresnel Scale (in km) divided by the event's velocity (in km/s), if the exposure time is shorter than this value the software will use 1/10 of the exposure time. The parameters of interest are, mostly, the immersion and emersion times. Moreover, the opacity of the object can also be of interest for particular cases (occultations by rings, for instance), where 1.0 means an opaque body and 0.0 a transparent one.

The fit consists into create models and compare them with the observational light curve, through a standard χ^2 test. The parameters that minimizes the χ^2 are the fitted parameters and their uncertainties are obtained from the values within $1\sigma = \chi^2_{min} + 1$ and $3\sigma = \chi^2_{min} + 9$. The result of the fit is a `ChiSquare` object, and its functions can be found at its specific Jupyter-Notebook.

[33]:
```
lc_1a.occ_model?
```

```
Signature:
lc_1a.occ_model(
    immersion_time,
    emersion_time,
    opacity,
    mask,
    npt_star=12,
    time_resolution_factor=10,
    flux_min=0,
    flux_max=1,
)
Docstring:
Returns the modelled light curve.

The modelled light curve takes into account the fresnel diffraction, the
star diameter and the instrumental response.

Parameters
```

(continues on next page)

(continued from previous page)

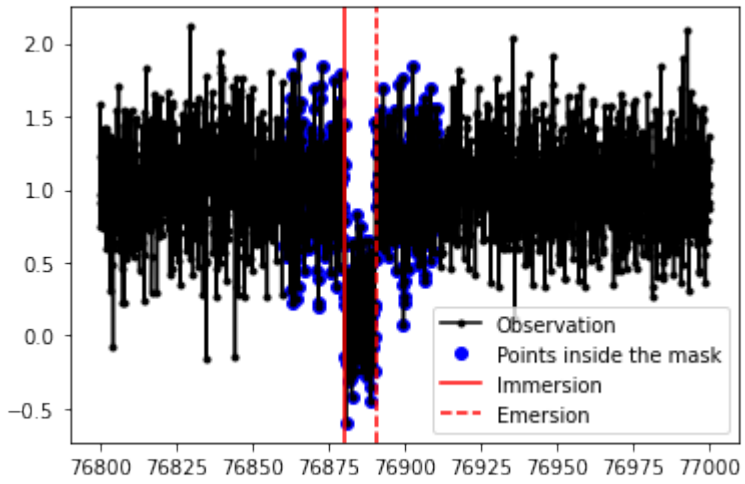
```
----------
immersion_time : `int`, `float`
    Immersion time, in seconds.

emersion_time : `int`, `float`
    Emersion time, in seconds.

opacity : `int`, `float`
    Opacity. Opaque = 1.0, transparent = 0.0,

mask : `bool` array
    Mask with True values to be computed.

npt_star : `int`, default=12
    Number of subdivisions for computing the star size effects.

time_resolution_factor : `int`, `float`, default: 10*fresnel scale
    Steps for fresnel scale used for modelling the light curve.

flux_min : `int`, `float`, default=0
    Bottom flux (only object).

flux_max : `int`, `float`, default=1
    Base flux (object plus star).
File:      ~/Documentos/códigos/SORA/sora/lightcurve/core.py
Type:      method
```

Let's find some parameters to create the model

[34]:
```
immersion_time = 76880.000  # seconds relative to tref
emersion_time  = 76891.000  # seconds relative to tref
opacity = 1.0            # 1.0 == opaque; 0.0 == transparent

tmin = immersion_time  - 20 # seconds relative to tref
tmax = emersion_time   + 20 # seconds relative to tref

mask = (lc_1a.time > tmin) & (lc_1a.time < tmax)
```

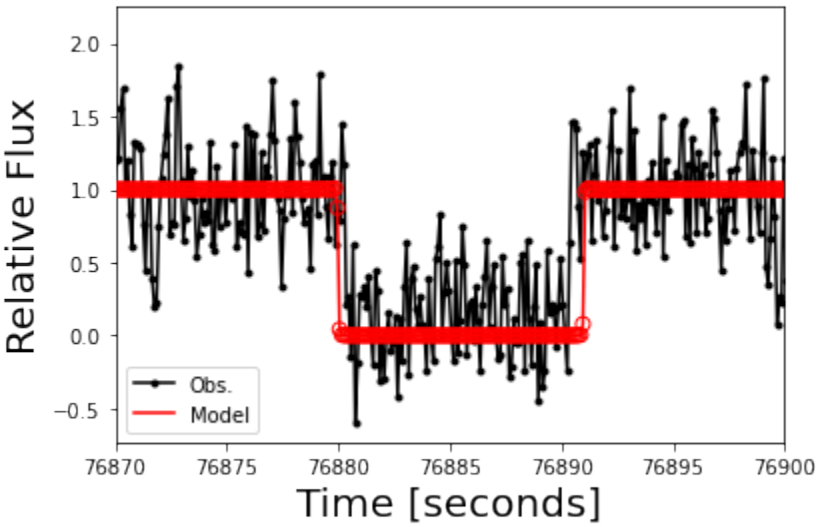
Let's give a look at the parameters we created (visual aid)

[35]:
```
pl.plot(lc_1a.time,lc_1a.flux,'k.-',zorder=1,label='Observation')
pl.plot(lc_1a.time[mask],lc_1a.flux[mask],'bo',zorder=0,label='Points inside the mask')
pl.axvline(immersion_time,color='r',linestyle='-',label='Immersion')
pl.axvline(emersion_time,color='r',linestyle='--',label='Emersion')
pl.legend()
pl.show()
```



Now, let's create the model

[36]:
```
lc_1a.occ_model(immersion_time, emersion_time, opacity, mask)

lc_1a.plot_lc()
pl.xlim(76870,76900)
pl.show()
```



Let's start with the fitting procedure

[37]:
```
lc_1a.occ_lcfit?
```

```
Signature: lc_1a.occ_lcfit(**kwargs)
Docstring:
Monte Carlo chi square fit for occultations lightcurve.

Parameters
----------
tmin : `int`, `float`
```

(continues on next page)

(continued from previous page)

```
    Minimum time to consider in the fit procedure, in seconds.

tmax : `int`, `float`
    Maximum time to consider in the fit procedure, in seconds.

flux_min : `int`, `float`, default=0
    Bottom flux (only object).

flux_max :`int`, `float`, default=1
    Base flux (object plus star).

immersion_time : `int`, `float`
    Initial guess for immersion time, in seconds.

emersion_time : `int`, `float`
    Initial guess for emersion time, in seconds.

opacity : `int`, `float`, default=1
    Initial guess for opacity. Opaque = 1, Transparent = 0.

delta_t : `int`, `float`
    Interval to fit immersion or emersion time.

dopacity : `int`, `float`, default=0
    Interval to fit opacity.

sigma : `int`, `float`, `array`, 'auto'
    Fluxes errors. If None it will use the `self.dflux`. If 'auto' it
    will calculate using the region outside the event.

loop : `int`, default=10000
    Number of tests to be done.

sigma_result : `int`, `float`
    Sigma value to be considered as result.

method : `str`, default=`chisqr`
    Method used to perform the fit. Available methods are:
    `chisqr` : monte carlo computation method used in versions of SORA <= 0.2.1.
    `fastchi` : monte carlo computation method, allows multithreading.
    `least_squares` or `ls`: best fit done used levenberg marquardt convergence␣
↪algorithm.
    `differential_evolution` or `de`: best fit done using genetic algorithms.
    All methods return a Chisquare object.

threads : `int`
    Number of threads/workers used to perform parallel computations of the chi square
    object. It works with all methods except `chisqr`, by default 1.

Returns
-------
chi2 : `sora.extra.ChiSquare`
```

(continues on next page)

(continued from previous page)

```
    ChiSquare object.
File:       ~/Documentos/códigos/SORA/sora/lightcurve/core.py
Type:       method
```

5.1. Automatic mode

This step is one of the most time consuming processes within the SORA package. It is expected that it would take between 1-10 minutes to complete 10000 loops. To spend less computational time with this tutorial, we will use only 1000 loops. However it is suggested that the user always set at least 10000 loops.

[38]:
```
lc_chi2 = lc_1a.occ_lcfit(loop=1000)
```

```
LightCurve fit: ||  - 100%
```

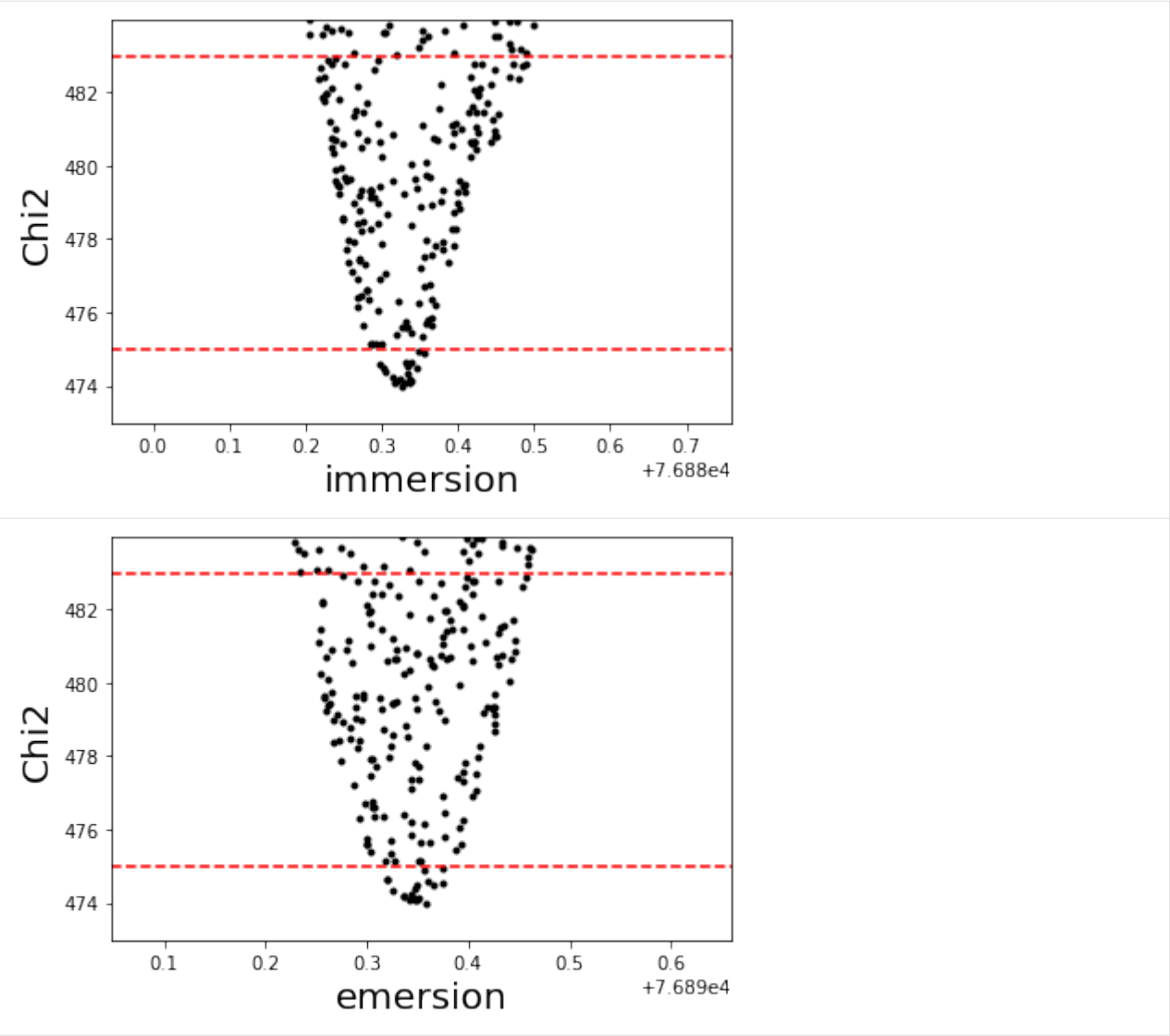
The result of the fitting is a ``ChiSquare`` Object

That is an auxiliary class, and its main information can be found at its specific Jupyter-Notebook, for the purpose of this tutorial only two main functions will be cover: `print(ChiSquare)` and `ChiSquare.plot_chi2()`.

[39]:
```
print(lc_chi2)
```

```
Minimum chi-square: 473.974
Number of fitted points: 496
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 0.959

immersion:
    1-sigma: 76880.326 +/- 0.029
    3-sigma: 76880.353 +/- 0.136

emersion:
    1-sigma: 76890.347 +/- 0.028
    3-sigma: 76890.354 +/- 0.102
```

[40]:
```
lc_chi2.plot_chi2()
```



These gaps within the chisquare curve are expected as we did a small number of loops (1000)

5.2. Fitting only the immersion

The user can set the parameters of choice to the fitting. That will allow using a more complex set of parameters for the model or do specific fits. One typical example is fitting one side of the chord (only the immersion time, for instance).

[41]:
```
immersion_time = 76880.338
tmin = 76873.000
tmax = 76887.000

mask = (lc_1b.time > tmin) & (lc_1b.time < tmax)
```

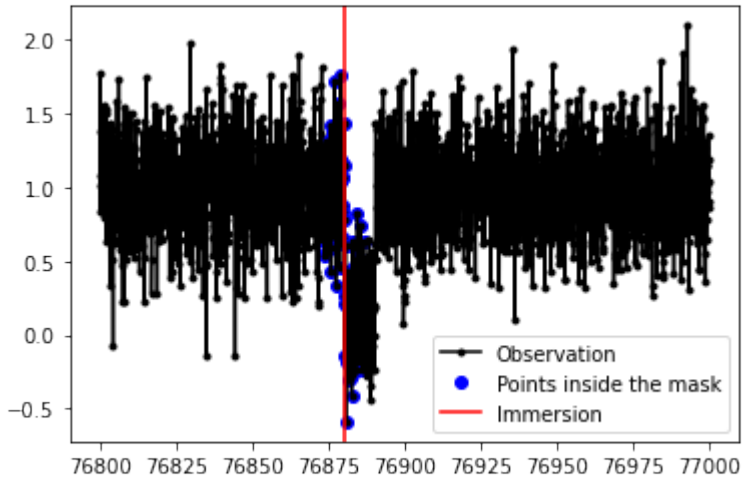
Let's give a look at the parameters we created (visual aid)

[42]:
```
pl.plot(lc_1b.time,lc_1b.flux,'k.-',zorder=1,label='Observation')
pl.plot(lc_1b.time[mask],lc_1b.flux[mask],'bo',zorder=0,label='Points inside the mask')
```

(continues on next page)

(continued from previous page)

```
pl.axvline(immersion_time,color='r',linestyle='-',label='Immersion')
pl.legend()
pl.show()
```



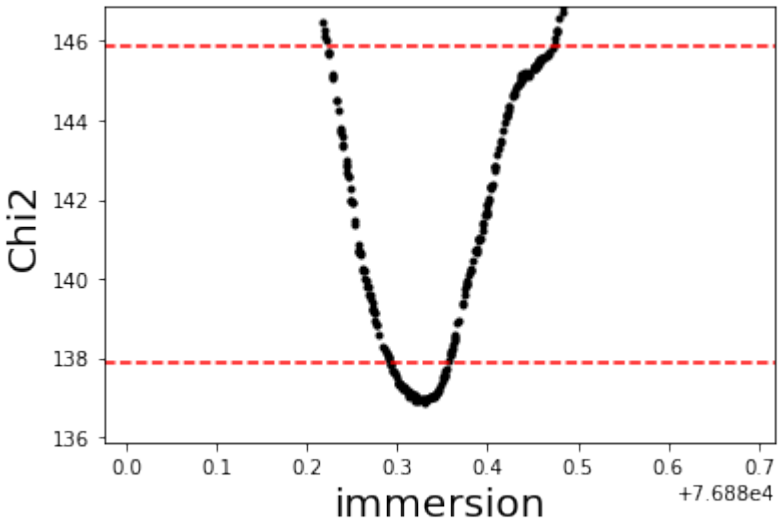
[43]:
```
lc_chi2_1b_imm = lc_1b.occ_lcfit(tmin=tmin, tmax=tmax, immersion_time=immersion_time,
                                 delta_t=0.5, loop=1000)
```

```
LightCurve fit: ||  - 100%
```

[44]:
```
print(lc_chi2_1b_imm)
```

```
Minimum chi-square: 136.864
Number of fitted points: 140
Number of fitted parameters: 1
Minimum chi-square per degree of freedom: 0.985

immersion:
    1-sigma: 76880.323 +/- 0.031
    3-sigma: 76880.347 +/- 0.124
```

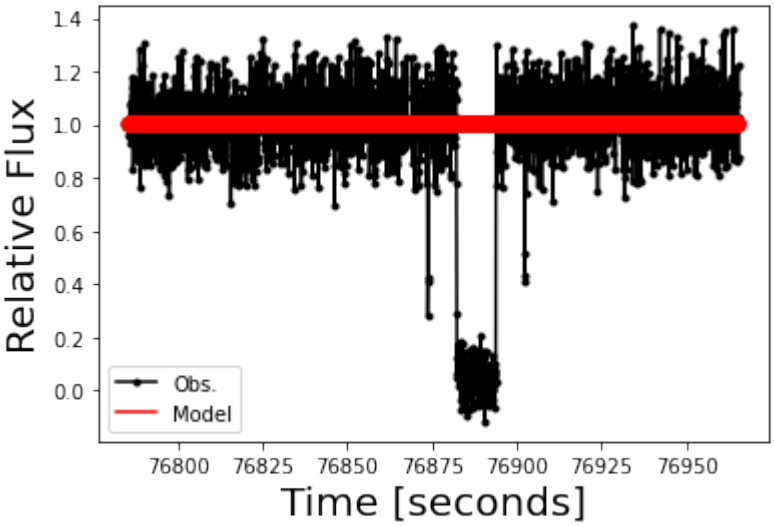
[45]:
```
lc_chi2_1b_imm.plot_chi2()
```



5.3. Fitting the times and the opacity

Let's give a look in another LightCurve.

[46]:
```
lc_2a.plot_lc()
```



In this light curve, aside the occultation by the main body there are two secondary events (rings), now let's determine the times and the opacity for the first detection.

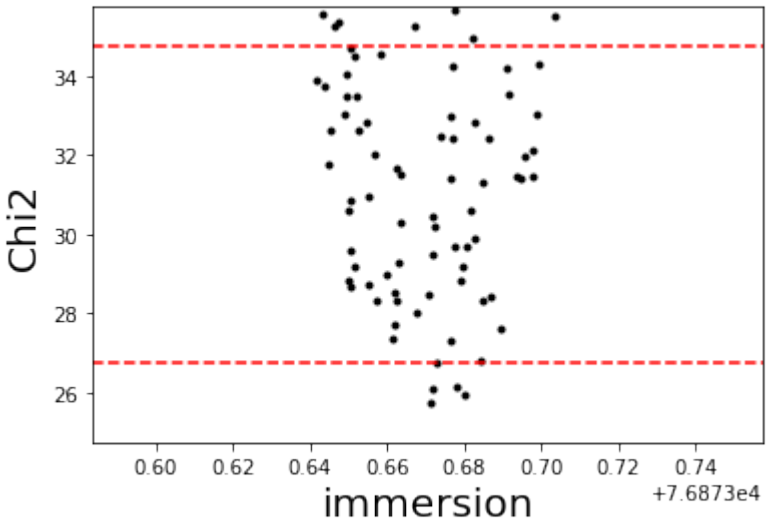
[47]:
```
lc_chi2_2a_ring = lc_2a.occ_lcfit(tmin=76872.676, tmax=76874.892, immersion_time=76873.
↪676,
                                  emersion_time=76873.892, opacity=0.50, delta_t=0.2,
                                  dopacity=0.5, loop=10000)
```

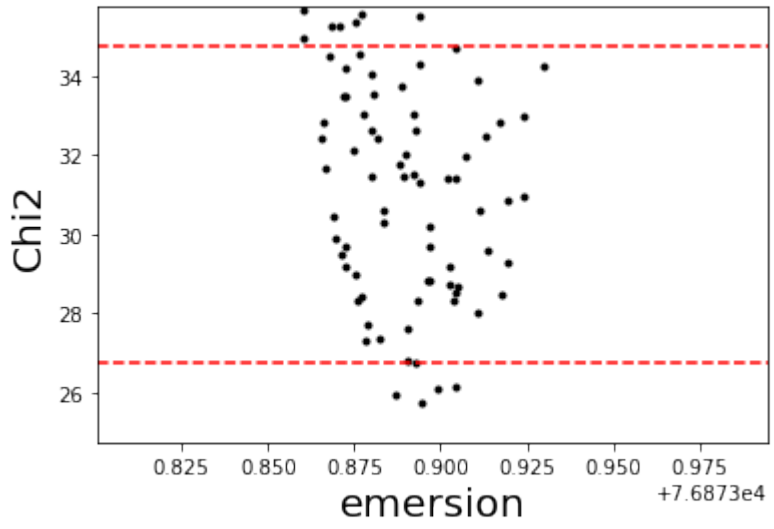
```
LightCurve fit: ||  - 100%
```

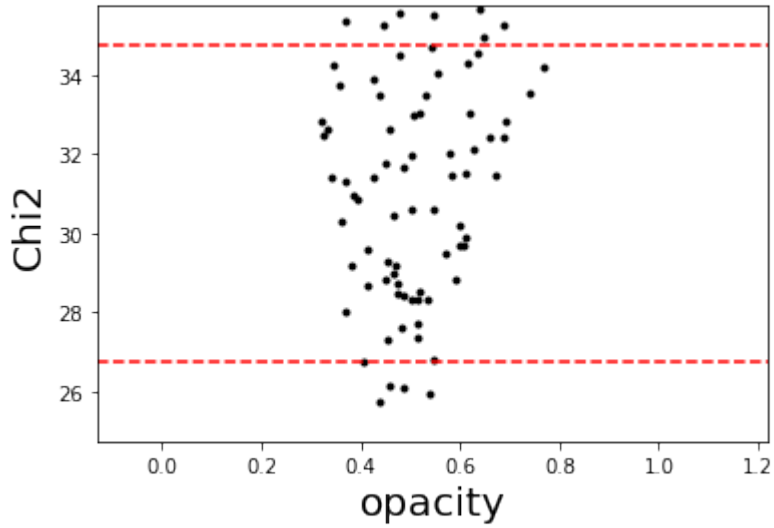
[48]:
```
print(lc_chi2_2a_ring)
```

```
Minimum chi-square: 25.746
Number of fitted points: 27
Number of fitted parameters: 3
Minimum chi-square per degree of freedom: 1.073

immersion:
    1-sigma: 76873.676 +/- 0.004
    3-sigma: 76873.670 +/- 0.029

emersion:
    1-sigma: 76873.896 +/- 0.009
    3-sigma: 76873.898 +/- 0.032

opacity:
    1-sigma: 0.488 +/- 0.052
    3-sigma: 0.544 +/- 0.225
```

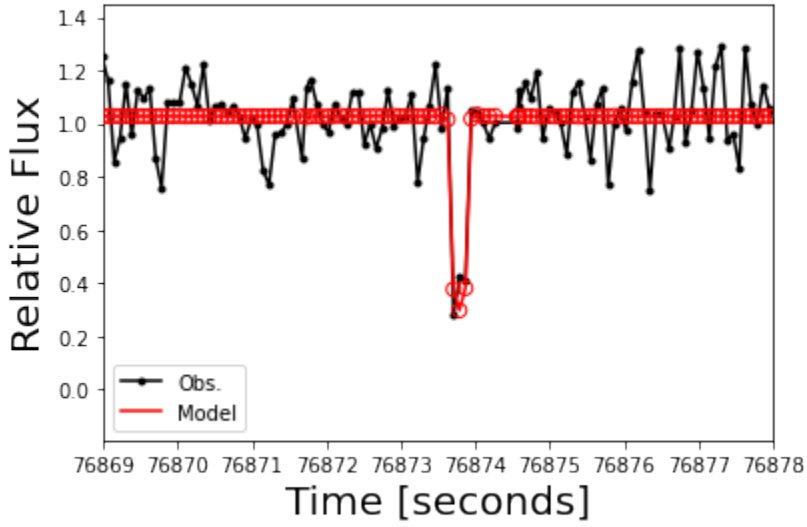
[49]:
```
lc_chi2_2a_ring.plot_chi2()
```







[50]:
```
lc_2a.plot_lc()
pl.xlim(76869,76878)
pl.legend()
pl.show()
```



9.5.7 6. Viewing and saving the results

At the end, the immersion and emersion times can be easily accessed using `LightCurve.immersion` and `LightCurve.emersion`. Their uncertainties can be found at `LightCurve.immersion_err` and `LightCurve.emersion_err`. Plot functions were created to facilitate the user to created the desired plots, an automatic plot can be created using `Light_curve.plot_lc()` and `LightCurve.plot_model()`, and the user can control the plot parameter using matplotlib functions. These plots can be saved using `matplotlib.pyplot.savefig()`.

[51]:
```
print('Imersion time {} UTC'.format(lc_1a.immersion))
print('Imersion time err: {:.3f} seconds'.format(lc_1a.immersion_err))
print('\n')
print('Emersion time {} UTC'.format(lc_1a.emersion))
print('Emersion time err: {:.3f} seconds'.format(lc_1a.emersion_err))
```

```
Imersion time 2017-06-22 21:21:20.326 UTC
Imersion time err: 0.029 seconds


Emersion time 2017-06-22 21:21:30.347 UTC
Emersion time err: 0.028 seconds
```

A complete log can be created by printing the ``LightCurve``

[52]:
```
print(lc_1a)
```

```
Light curve name: Example 1a
Initial time: 2017-06-22 21:20:00.056 UTC
End time:     2017-06-22 21:23:19.958 UTC
Duration:     3.332 minutes
Time offset:  0.000 seconds

Exposure time:    0.1000 seconds
Cycle time:       0.1002 seconds
Num. data points: 2000
```

(continues on next page)

(continued from previous page)

```
Bandpass:              0.700 +/- 0.300 microns
Object Distance:       15.00 AU
Used shadow velocity: 22.000 km/s
Fresnel scale:         0.040 seconds or 0.88 km
Stellar size effect:   0.009 seconds or 0.20 km
Inst. response:        0.100 seconds or 2.20 km
Dead time effect:      0.000 seconds or 0.00 km
Model resolution:      0.004 seconds or 0.09 km
Modelled baseflux:     1.029
Modelled bottomflux:   0.109
Light curve sigma:     0.336

Immersion time: 2017-06-22 21:21:20.326 UTC +/- 0.029 seconds
Emersion time:  2017-06-22 21:21:30.347 UTC +/- 0.028 seconds

Monte Carlo chi square fit.

Minimum chi-square: 473.974
Number of fitted points: 496
Number of fitted parameters: 2
Minimum chi-square per degree of freedom: 0.959

immersion:
    1-sigma: 76880.326 +/- 0.029
    3-sigma: 76880.353 +/- 0.136

emersion:
    1-sigma: 76890.347 +/- 0.028
    3-sigma: 76890.354 +/- 0.102
```

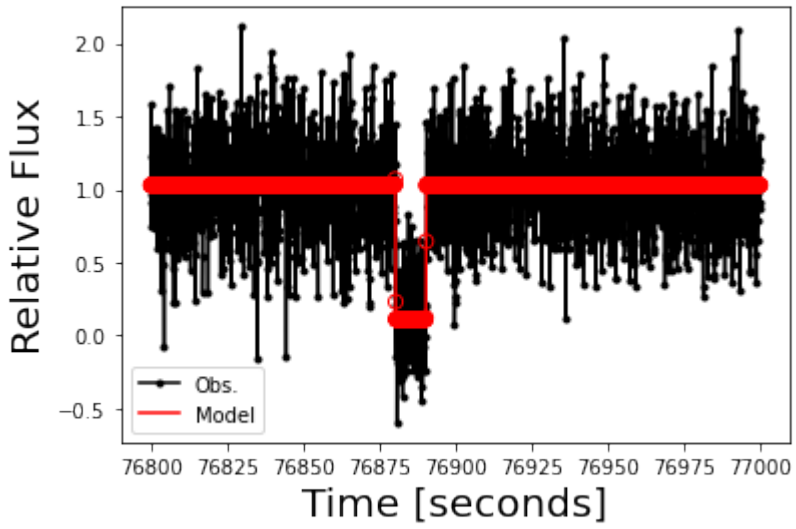
And this log can be saved as a file using LightCurve.to_log()

[53]:
```
lc_1a.to_log('output/LC_test_1a.log')

!ls output/*.log
```

```
output/LC_test_1a.log
```

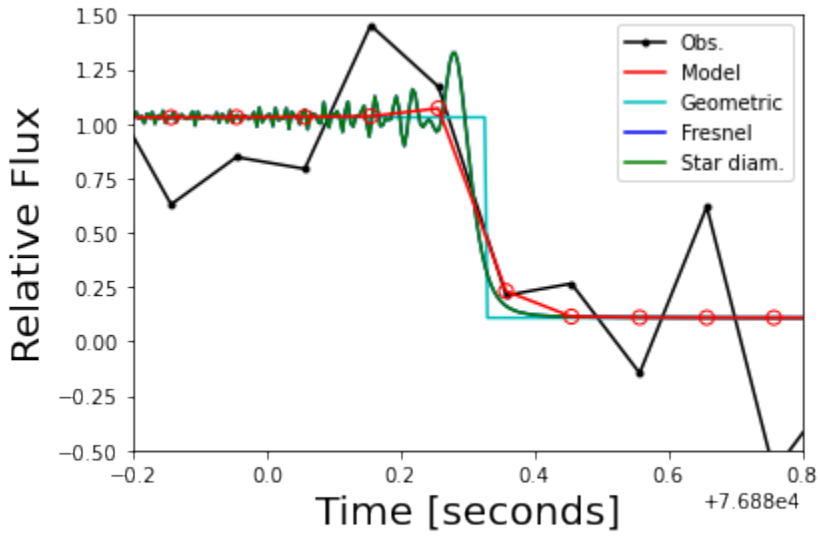
The post fitted light curve can be plotted using the `LightCurve.plot_lc()`.

[54]:
```
lc_1a.plot_lc()
```



The post fitted light curve complete model can be plotted using the `LightCurve.plot_model()`

[55]:
```
lc_1a.plot_lc()
lc_1a.plot_model()

pl.xlim(76879.8,76880.8)
pl.ylim(-0.5,1.5)

pl.show()
```



The `LightCurve` data values can be saved using `LightCurve.to_file()`.

If the model was correctly fitted with SORA, 4 files should be created: (i) One containing the observational light curve and its convoluted model; (ii) Containing the complete light curve model (geometric, affected by fresnel diffraction and by the star apparent diameter); and (iii and iv) with the labels for each colunm for other two files. Otherwise only 2 files will be created: the observational one and its labels.

[56]:
```
lc_1a.to_file?
```

```
Signature: lc_1a.to_file(namefile=None)
Docstring:
Saves the light curve to a file.

Parameters
----------
namefile : `str`
    Filename to save the data.
File:      ~/Documentos/códigos/SORA/sora/lightcurve/core.py
Type:      method
```

[57]:
```
lc_1a.to_file(namefile='output/LC_test_1a.dat')

!ls output/*.dat
print('\n')
!ls output/*.dat.label
```

```
output/example_chi2.dat  output/LC_test_1a.dat       output/model_LC_test_1a.dat


output/example_chi2.dat.label  output/model_LC_test_1a.dat.label
output/LC_test_1a.dat.label
```

One last thing, a common issue within the stellar occultation light curves is a time offset for a specific light curve

To fit the body size and shape the user should consider the times (immersion and emersion) for all the observers. It is not unusual to have the different chords projected at the sky plane not be appropriately 'aligned'. Those shifts are often caused by instrumental issues (lack of GPS, software's features, internet protocols, among many others). To deal with this issue, the user can add a time offset (in seconds) manually at the `LightCurve` Object using the `LightCurve.dt`. Then the SORA package will automatically consider this offset to the fitted (or instantiated) parameters.

[58]:
```
lc_1a.dt = 0 #seconds

print('Imersion time {} UTC'.format(lc_1a.immersion))

lc_1a.dt = 30 #seconds

print('Imersion time {} UTC'.format(lc_1a.immersion))
```

```
Imersion time 2017-06-22 21:21:20.326 UTC
Imersion time 2017-06-22 21:21:50.326 UTC
```

This Jupyter-Notebook was designed as a tutorial for how to work with the `LightCurve` Class. More information about the other classes, please refer to their specif Jupyter-Notebook. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The End

9.6 Occultation Class

The `Occultation` Class within SORA was created to reduce and analyze stellar occultations, and manage all the other Objects in a SORA occultation. Here we have some useful tasks that allow converting the times for each observatory in positions in the sky plane (ξ, η), fit an ellipse to the points, obtain the astrometrical position resulting, among others.

The documentation here contains the details about every step.

This Jupyter-Notebook was designed as a tutorial for how to work with the `Occultation` Class. More information about the other classes, please refer to their specif Jupyter-Notebook. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The `Occultation` Docstring was designed to help the users. Also, each function has its Docstring containing its main purpose and the needed parameters (physical description and formats). Please, do not hesitate to use it.

9.6.1 0. Index

1. *Instantiating an Occultation Object and adding observations*
2. *Projecting the times in the sky plane and the Chords*
3. *Ellipse fit*
4. *Viewing and saving the results*

[1]:
```
## SORA package
from sora import Occultation, Body, Star, LightCurve, Observer

## Other main packages
#from astropy.time import Time
import astropy.units as u

## Usual packages
import numpy as np
import matplotlib.pylab as pl
import os
```

```
SORA version: 0.3
```

9.6.2 1. Instantiating an Occultation Object and adding observations

The `Occultation` Class can be instantiated in only one way. For this it is needed a `Star` Object, an `Body` Object with an `Ephemeris` Object (`EphemKernel`, `EphemPlanet` or `EphemHorizons`), and the occultation time (within 60 minutes of the correct value).

[2]:
```
Occultation?
```

```
Init signature:
Occultation(
    star,
    body=None,
    ephem=None,
    time=None,
    reference_center='geocenter',
)
```

(continues on next page)

(continued from previous page)

```
Docstring:
Instantiates the Occultation object and performs the reduction of the
occultation.

Attributes
----------
star : `sora.Star`, `str`, required
    the coordinate of the star in the same reference frame as the ephemeris.
    It must be a Star object or a string with the coordinates of the object
    to search on Vizier.

body : `sora.Body`, `str`
    Object that will occult the star. It must be a Body object or its name
    to search in the Small Body Database.

ephem : `sora.Ephem`, `list`
    Object ephemeris. It must be an Ephemeris object or a list.

time : `str`, `astropy.time.Time`, required
    Reference time of the occultation. Time does not need to be exact, but
    needs to be within approximately 50 minutes of the occultation closest
    approach to calculate occultation parameters.

reference_center : `str`, `sora.Observer`, `sora.Spacecraft`
    A SORA observer object or a string 'geocenter'.
    The occultation parameters will be calculated in respect
    to this reference as center of projection.


Important
---------
When instantiating with "body" and "ephem", the user may define the
Occultation in 3 ways:

1. With `body` and `ephem`.

2. With only "body". In this case, the "body" parameter must be a Body
object and have an ephemeris associated (see Body documentation).

3. With only `ephem`. In this case, the `ephem` parameter must be one of the
Ephem Classes and have a name (see Ephem documentation) to search for the
body in the Small Body Database.
File:           ~/Documentos/códigos/SORA/sora/occultation/core.py
Type:           type
Subclasses:
```

First let's instantiate the ``Star`` and the ``Body``

SORA will automatically search for the star information. A warning will raise when any information is missing. In this example, there is no star radius available in Gaia.

[3]:
```
chariklo = Body(name='Chariklo',
                ephem=['input/bsp/Chariklo.bsp', 'input/bsp/de438_small.bsp'])

star_occ  = Star(coord='18 55 15.65250 -31 31 21.67051') # Occ Chariklo 22-06-2017
```

```
Obtaining data for Chariklo from SBDB
1 GaiaDR3 star found band={'G': 14.223702}
star coordinate at J2016.0: RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.
↪018 mas

Downloading star parameters from I/297/out
```

Now, we can instantiate the `Occultation`

[4]:
```
occ = Occultation(star=star_occ, body=chariklo, time='2017-06-22 21:18')

print(occ)
```

```
Stellar occultation of star GaiaDR3 6760223758801661440 by 10199 Chariklo (1997 CU26).

Geocentric Closest Approach: 0.049 arcsec
Instant of CA: 2017-06-22 21:18:48.200
Position Angle: 359.72 deg
Geocentric shadow velocity: -22.00 km / s
Sun-Geocenter-Target angle:  166.42 deg
Moon-Geocenter-Target angle: 149.11 deg


No observations reported

################################################################################
                                    STAR
################################################################################
GaiaDR3 star Source ID: 6760223758801661440
ICRS star coordinate at J2016.0:
RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.0180 mas
pmRA=3.556 +/- 0.025 mas/yr, pmDEC=-2.050 +/- 0.020 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.2121 +/- 0.0228 mas, Rad. Vel.=-40.49 +/- 3.73 km/s

Magnitudes: G: 14.224, B: 14.320, V: 13.530, R: 14.180, J: 12.395, H: 11.781,
            K: 11.627

Apparent diameter from Kervella et. al (2004):
    V: 0.0216 mas, B: 0.0216 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0238 mas, V: 0.0244 mas
    ms: B: 0.0261 mas, V: 0.0198 mas
    vs: B: 0.0350 mas, V: 0.0315 mas

Geocentric star coordinate at occultation Epoch (2017-06-22 21:18:48.200):
RA=18h55m15.65251s +/- 0.0327 mas, DEC=-31d31m21.6706s +/- 0.0341 mas

################################################################################
```

(continues on next page)

(continued from previous page)

```
                          10199 Chariklo (1997 CU26)
################################################################################
Object Orbital Class: Centaur
Spectral Type:
    SMASS: D  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Relatively featureless spectrum with very steep red slope.
Discovered 1997-Feb-15 by Spacewatch at Kitt Peak

Physical parameters:
Diameter:
    302 +/- 30 km
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),
Rotation:
    7.004 +/- 0 h
    Reference: LCDB (Rev. 2021-June); Warner et al., 2009, [Result based on less than␣
↪full coverage, so that the period may be wrong by 30 percent or so.]  REFERENCE LIST:
↪[Fornasier, S.; Lazzaro, D.; Alvarez-Candal, A.; Snodgrass, C.; et al. (2014) Astron.␣
↪Astrophys. 568, L11.], [Leiva, R.; Sicardy, B.; Camargo, J.I.B.; Desmars, J.; et al.␣
↪(2017) Astron. J. 154, A159.]
Absolute Magnitude:
    6.54 +/- 0 mag
    Reference: MPO691682,
Albedo:
    0.045 +/- 0.01
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),

Ellipsoid: 151.0 x 151.0 x 151.0

----------- Ephemeris -----------

EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec
```

If the occultation was observed by a spacecraft, the Occultation parameters should be refered to this observer. For that, it is necessary to pass the observer as the center of reference for the `Occultation`

[5]:
```
# from sora import Spacecraft
# nh = Spacecraft(name='New Horizons', spkid='-98', ephem='horizons')
# nh_occ = Occultation(star=star, body=body, time=time, reference_center=nh)
```

Note that at the `print(Occultation)` there are no observations added to this `Occultation` yet.

It is needed one `Observer` and one `LightCurve` to define an `Chord`.

Let's instanciate the `Observers`

Here we will give 5 locations

[6]:
```
out = Observer(name='Outeniqua'  ,lon='+16 49 17.710', lat='-21 17 58.170', height =1416)
ond = Observer(name='Onduruquea' ,lon='+15 59 33.750', lat='-21 36 26.040', height =1220)
```

(continues on next page)

(continued from previous page)

```
tiv = Observer(name='Tivoli'     ,lon='+18 01 01.240', lat='-23 27 40.190', height =1344)
whc = Observer(name='Windhoek'   ,lon='+17 06 31.900', lat='-22 41 55.160', height =1902)
hak = Observer(name='Hakos'      ,lon='+16 21 41.320', lat='-23 14 11.040', height =1843)
```

If a spacecraft observed the occultation, a `Spacecraft` object could be defined and added to the `Occultation` normally.

Let's instanciate the `LightCurves`

Here we give 6 light curves. Note that 2 are for the same site observed with different telescopes.

[7]:
```
#THE ERRORS ARE INCREASED FOR BETTER VISUALIZATION

out_lc = LightCurve(name='Outeniqua lc',
                    initial_time='2017-06-22 21:20:00.056',
                    end_time ='2017-06-22 21:29:59.963',
                    immersion='2017-06-22 21:21:20.329',immersion_err=0.320,
                    emersion ='2017-06-22 21:21:30.343',emersion_err=0.340)

ond_lc = LightCurve(name='Onduruquea lc',
                    initial_time='2017-06-22 21:11:52.175',
                    end_time ='2017-06-22 21:25:13.389',
                    immersion='2017-06-22 21:21:22.213',immersion_err=0.100,
                    emersion ='2017-06-22 21:21:33.824',emersion_err=0.110)

tiv_lc = LightCurve(name='Tivoli lc',
                    initial_time='2017-06-22 21:16:00.094',
                    end_time ='2017-06-22 21:28:00.018',
                    immersion='2017-06-22 21:21:15.628',immersion_err=0.700,
                    emersion ='2017-06-22 21:21:19.988',emersion_err=0.700)

whc_c14_lc = LightCurve(name='Windhoek C14 lc',
                    initial_time='2017-06-22 21:12:48.250',
                    end_time ='2017-06-22 21:32:47.963',
                    immersion='2017-06-22 21:21:17.609',immersion_err=0.240,
                    emersion ='2017-06-22 21:21:27.564',emersion_err=0.260)

whc_d16_lc = LightCurve(name='Windhoek D16 lc',
                    initial_time='2017-06-22 21:20:01.884',
                    end_time ='2017-06-22 21:22:21.894',
                    immersion='2017-06-22 21:21:17.288',immersion_err=0.280,
                    emersion ='2017-06-22 21:21:27.228',emersion_err=0.340)

hak_lc = LightCurve(name='Hakos lc',
                    initial_time='2017-06-22 21:10:19.461',
                    end_time ='2017-06-22 21:30:19.345')
```

To add the observation to the `Occultation` **just use the** `Occultation.chords.add_chord()`

During this step the `LightCurve` model parameters are automaticaly updated using the `Occultation` parameters: the Shadow's velocity, the star diameter and the distance to the occulting body. This means that is no need for the user to do the `LightCurve.set_dist()`, `LightCurve.set_vel()` and `LightCurve.set_star_diam()`.

[8]:
```
occ.chords.add_chord(observer=out, lightcurve=out_lc)
occ.chords.add_chord(observer=ond, lightcurve=ond_lc)
occ.chords.add_chord(observer=tiv, lightcurve=tiv_lc)
```

```
/home/altair/Documentos/códigos/SORA/sora/body/core.py:332: UserWarning: H and/or G is␣
↪not defined for 10199 Chariklo. Searching into JPL Horizons service
  warnings.warn('H and/or G is not defined for {}. Searching into JPL Horizons service'.
↪format(self.shortname))
```

[8]:
```
<Chord: Tivoli>
```

Note that a warning comes up if the flux drop is not calculated automatically.

The same goes for two observation at with the same `Observer` , however is important to define their names as different values

[9]:
```
occ.chords.add_chord(name='Windhoek C14', observer=whc, lightcurve=whc_c14_lc)
occ.chords.add_chord(name='Windhoek D16', observer=whc, lightcurve=whc_d16_lc)
```

[9]:
```
<Chord: Windhoek D16>
```

Also, the same goes for negative observations

[10]:
```
occ.chords.add_chord(observer=hak, lightcurve=hak_lc)
```

[10]:
```
<Chord: Hakos>
```

If an spacecraft observed the occultation, just define the `Spacecraft` object and add it to the `Occutation` as usual.

[11]:
```
# from sora import Spacecraft
# nh = Spacecraft(name='New Horizons', spkid='-98', ephem='horizons')
# occ.chords.add_chord(observer=nh, lightcurve=lightcurve)
```

To check which observers were added to the `Occultation` just use `Occultation.chords`

[12]:
```
occ.chords
```

[12]:
```
<ChordList:
    0: Chord(Outeniqua)
    1: Chord(Onduruquea)
    2: Chord(Tivoli)
    3: Chord(Windhoek C14)
    4: Chord(Windhoek D16)
    5: Chord(Hakos)>
```

If any of them was wrongly added just remove it using `Occultation.chords.remove_chord()`

[13]:
```
occ.chords.remove_chord(name='Outeniqua')
```

[14]:
```
occ.chords
```

[14]:
```
<ChordList:
    0: Chord(Onduruquea)
    1: Chord(Tivoli)
    2: Chord(Windhoek C14)
    3: Chord(Windhoek D16)
    4: Chord(Hakos)>
```

If all the chords were wrongly added the user can clear the `Occultation` using `Occultation.chords.clear()`

[15]:
```
occ.chords.clear()
```

[16]:
```
occ.chords
```

[16]:
```
<ChordList:>
```

[17]:
```
occ.chords.add_chord(observer=out, lightcurve=out_lc)
occ.chords.add_chord(observer=ond, lightcurve=ond_lc)
occ.chords.add_chord(observer=tiv, lightcurve=tiv_lc)
occ.chords.add_chord(name='Windhoek C14', observer=whc, lightcurve=whc_c14_lc)
occ.chords.add_chord(name='Windhoek D16', observer=whc, lightcurve=whc_d16_lc)
occ.chords.add_chord(observer=hak, lightcurve=hak_lc)
```

[17]:
```
<Chord: Hakos>
```

9.6.3 2. Projecting the times in the sky plane and the Chords

This step is done automatically without the user having to ask for it. The user can see it automatically using `Occultation.chords.summary()` that contains all positions for all observations added.

[18]:
```
occ.chords.summary()
```

```
    Name      Longitude      Latitude      status              time               f       
↪ g
------------ ------------ ------------- ------------ ----------------------- ---------- --
↪-----
   Outeniqua 16d49m17.71s -21d17m58.17s Initial Time 2017-06-22 21:20:00.056  -1912.82  
↪45.76
                                        Immersion 2017-06-22 21:21:20.329   -118.28  
↪65.39
                                         Emersion 2017-06-22 21:21:30.343    105.60  
↪67.83
                                         End Time 2017-06-22 21:29:59.963  11501.40 
↪190.23
  Onduruquea 15d59m33.75s -21d36m26.04s Initial Time 2017-06-22 21:11:52.175 -12876.38 -
↪135.47
                                        Immersion 2017-06-22 21:21:22.213   -138.25  
↪ 7.19
                                         Emersion 2017-06-22 21:21:33.824    121.28  
↪10.05
                                         End Time 2017-06-22 21:25:13.389   5029.49  
↪63.76
      Tivoli 18d01m01.24s -23d27m40.19s Initial Time 2017-06-22 21:16:00.094  -7123.66 -
↪202.63
                                        Immersion 2017-06-22 21:21:15.628    -70.56 -
↪126.77
                                         Emersion 2017-06-22 21:21:19.988     26.91 -
↪125.73
                                         End Time 2017-06-22 21:28:00.018   8971.45  -
↪31.61
Windhoek C14  17d06m31.9s -22d41m55.16s Initial Time 2017-06-22 21:12:48.250 -11505.42 -
```

(continues on next page)

(continued from previous page)

```
↪198.50
                                          Immersion 2017-06-22 21:21:17.609   -121.72  -
↪73.52
                                           Emersion 2017-06-22 21:21:27.564    100.82  -
↪71.11
                                           End Time 2017-06-22 21:32:47.963  15314.98  ␣
↪90.02
Windhoek D16  17d06m31.9s -22d41m55.16s Initial Time 2017-06-22 21:20:01.884  -1814.42  -
↪91.87
                                          Immersion 2017-06-22 21:21:17.288   -128.89  -
↪73.59
                                           Emersion 2017-06-22 21:21:27.228     93.31  -
↪71.19
                                           End Time 2017-06-22 21:22:21.894   1315.35  -
↪58.00
      Hakos 16d21m41.32s -23d14m11.04s Initial Time 2017-06-22 21:10:19.461 -14875.34 -
↪316.83
                                           End Time 2017-06-22 21:30:19.345  11939.87  -
↪23.52
```

Let's give a look to an specific `Chord`

[19]:
```
chord = occ.chords['Outeniqua']
```

The user can easily acess the f and g for the immersion and emersion times using the `Chord.get_fg()`

[20]:
```
chord.get_fg()
```

[20]:
```
(array([-118.27875072,  105.59793973]), array([65.3915666 , 67.83395663]))
```

To see the f and g projection for the immersion and emersion times used in the fitting process, the user can use the following function

[21]:
```
names, fg, error = occ.chords.get_limb_points()
print(names)
print(fg)
print(error)
```

```
['Outeniqua_immersion' 'Outeniqua_emersion' 'Onduruquea_immersion'
 'Onduruquea_emersion' 'Tivoli_immersion' 'Tivoli_emersion'
 'Windhoek C14_immersion' 'Windhoek C14_emersion' 'Windhoek D16_immersion'
 'Windhoek D16_emersion']
[[-118.27875072   65.3915666 ]
 [ 105.59793973   67.83395663]
 [-138.25381491    7.1933912 ]
 [ 121.275169     10.05237144]
 [ -70.56164003 -126.77190052]
 [  26.91022935 -125.73348183]
 [-121.71711407  -73.51645459]
 [ 100.81847461  -71.10987823]
 [-128.89276574  -73.59407832]
 [  93.30744752  -71.19108221]]
[[ 7.15400526  0.07806865]
 [ 7.60120413  0.08290053]
```

(continues on next page)

(continued from previous page)

```
 [ 2.23518846  0.02463189]
 [ 2.45872898  0.0270767 ]
 [15.64912805  0.16673645]
 [15.64918323  0.16669294]
 [ 5.36497525  0.05803441]
 [ 5.81210573  0.06283513]
 [ 6.25913618  0.06770901]
 [ 7.60044325  0.08217062]]
```

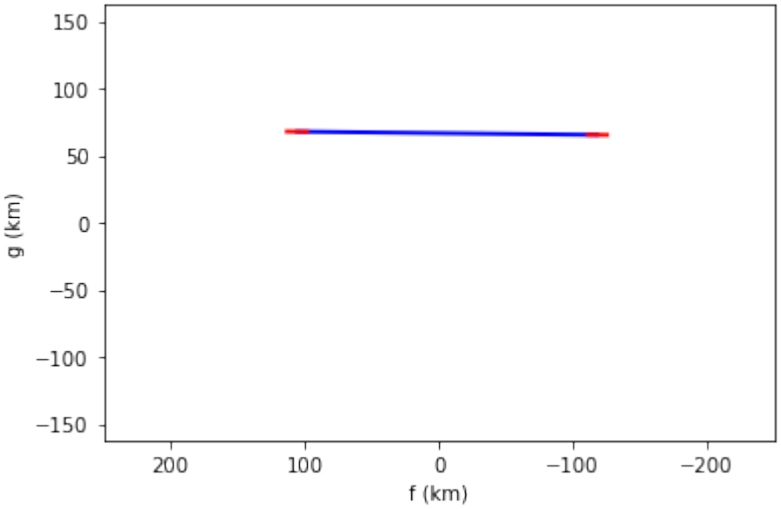
The user can also set a specific time

[22]:
```
chord.get_fg(time='2017-06-22 21:00:00.000')
```

[22]:
```
(-28724.98453460884, -258.35753813600354)
```

Finally, the user can visually see the chord in the Sky-plane using the `Chord.plot_chord()`.

[23]:
```
chord.plot_chord(segment='positive', color='blue')
chord.plot_chord(segment='error', color='red')

pl.xlim(+250,-250)
pl.ylim(-250,+250)
pl.show()
```



Let's define a `LightCurve` **with times and fluxes**

[24]:
```
chord2 = occ.chords['Hakos']

chord2.lightcurve.set_flux(file='input/lightcurves/lc_example_5.dat', exptime=1.000,␣
↪usecols=[0,1])
```

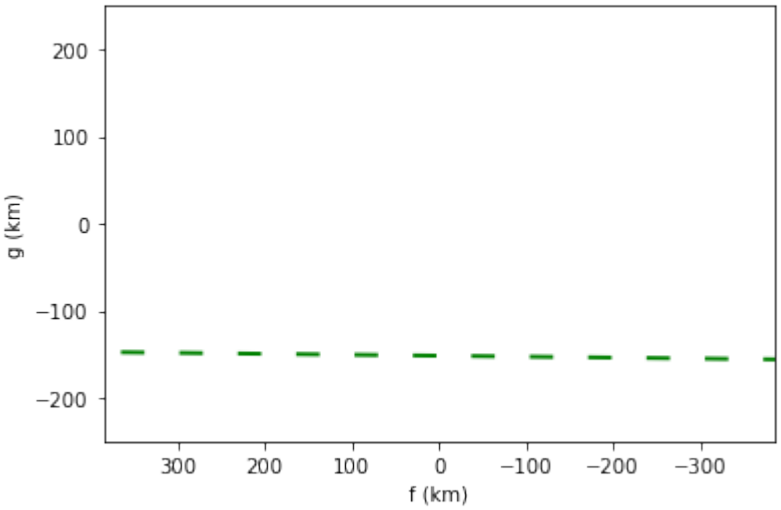
The `Chord.plot_chord()` **can also be used with an** `linestyle = "exposure"` **to see the times where the the chord in fact holds information and the readout time (without information).**

[25]:
```
chord2.plot_chord(segment='negative', linestyle='exposure', color='green')
```

(continues on next page)

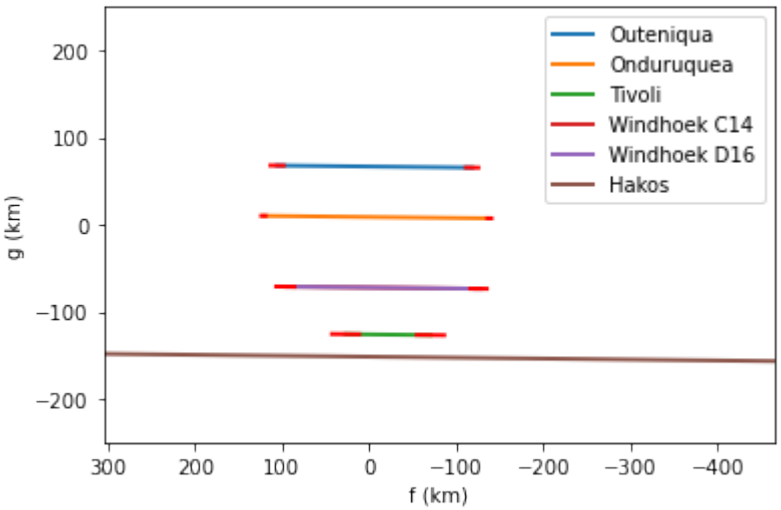
(continued from previous page)

```
pl.xlim(+250,-250)
pl.ylim(-250,+250)
pl.show()
```



Now let's consider all the chords

[26]:
```
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```



If there are known time shifts, this can be easily solved using `LightCurve.dt`

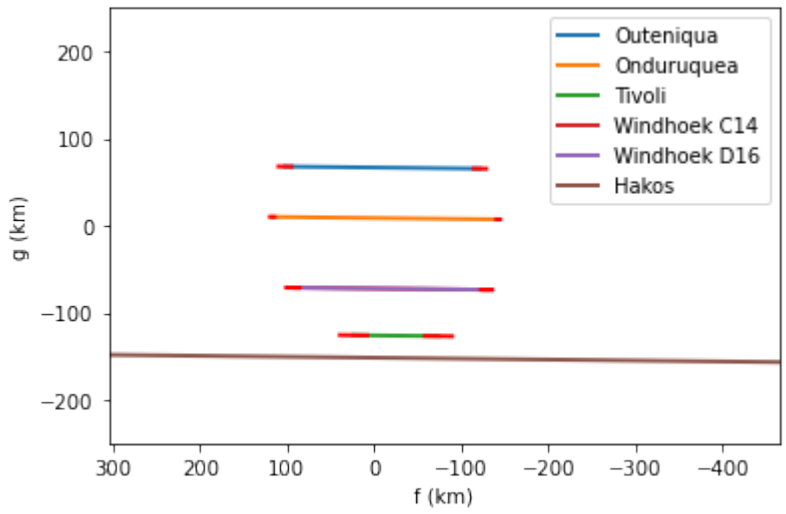
[27]:
```
out_lc.dt = -0.150
ond_lc.dt = -0.190
```

(continues on next page)

(continued from previous page)

```
tiv_lc.dt = -0.150
whc_c14_lc.dt = -0.375
whc_d16_lc.dt = +0.000
```

[28]:
```
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```



The user can save these positions

[29]:
```
occ.to_file()
```

9.6.4 3. Ellipse fit

The next step is the fitting of an ellipse. The five parameters that describe the ellipse are:

1. and **2.** The centre position (f_0, g_0)

3. The apparent equatorial radius, semi-major axix (a')

4. The oblatness ($\epsilon' = \frac{a'-b'}{a'}$)

5. The position angle of the pole, semi-minor axis (P)

The result of the fit is a `ChiSquare` Object, and its functions can be found at its specific Jupyter-Notebook.

Here, there is only the manual method, the user should provide the parameters to the fit and also the region for searching each parameters.

The equation to be minimize is:

$\chi^2 = \sum_i^N \frac{(r_i - r'_i)^2}{\sigma_i^2 + \sigma_{model}^2}$

where: - r_i is the radial distance between the i^{th} observed point and the ellipse centre; - r'_i is the radial distance between the modelled ellipse's i^{th} point and the ellipse centre; - σ_i is the uncertainty of the i^{th} observed point - σ_{model} is the model uncertainty, that is releated to the difference between the real apparent shape of the occultating object and the ellipse model.

[30]:
```
occ.fit_ellipse?
```

```
Signature: occ.fit_ellipse(**kwargs)
Docstring:
Fits an ellipse to given occultation using given parameters.

Parameters
----------
center_f : `int`, `float`, default=0
    The coordinate in f of the ellipse center.

center_g : `int`, `float`, default=0
    The coordinate in g of the ellipse center.

equatorial_radius : `int`, `float`
    The Equatorial radius (semi-major axis) of the ellipse.

oblateness : `int`, `float`, default=0
    The oblateness of the ellipse.

position_angle : `int`, `float`, default=0
    The pole position angle of the ellipse in degrees.
    Zero is in the North direction ('g-positive'). Positive clockwise.

dcenter_f : `int`, `float`
    Interval for coordinate f of the ellipse center.

dcenter_g : `int`, `float`
    Interval for coordinate g of the ellipse center.

dequatorial_radius `int`, `float`
    Interval for the Equatorial radius (semi-major axis) of the ellipse.

doblateness : `int`, `float`
    Interval for the oblateness of the ellipse

dposition_angle : `int`, `float`
    Interval for the pole position angle of the ellipse in degrees.

loop : `int`, default=10000000
    The number of ellipses to attempt fitting.

dchi_min : `int`, `float`
    If given, it will only save ellipsis which chi square are smaller than
    chi_min + dchi_min. By default `None` when used with `method='chisqr`, and
    `3` for other methods.

number_chi : `int`, default=10000
    In the `chisqr` method if `dchi_min` is given, the procedure is repeated until
```

(continues on next page)

(continued from previous page)

```
    `number_chi` is reached.
    In other methods it is the number of values (simulations) that should lie within
    the provided `sigma_result`.

verbose : `bool`, default=False
    If True, it prints information while fitting.

ellipse_error : `int`, `float`
    Model uncertainty to be considered in the fit, in km.

sigma_result : `int`, `float`
    Sigma value to be considered as result.

method : `str`, default=`least_squares`
    Method used to perform the fit. Available methods are:
    `chisqr` : monte carlo computation method used in versions of SORA <= 0.2.1.
    `fastchi` : monte carlo computation method, allows multithreading .
    `least_squares` or `ls`: best fit done used levenberg marquardt convergence␣
↪algorithm.
    `differential_evolution` or ``de`: best fit done using genetic algorithms.
    All methods return a Chisquare object.

threads : `int`
    Number of threads/workers used to perform parallel computations of the chi square
    object. It works with all methods except `chisqr`, by default 1.

Returns
-------
chisquare : `sora.ChiSquare`
    A ChiSquare object with all parameters.

Important
---------
Each occultation is added as the first argument(s) directly.

Mandatory input parameters: 'center_f', 'center_g', 'equatorial_radius',
'oblateness', and 'position_angle'.

Parameters fitting interval: 'dcenter_f', 'dcenter_g', 'dequatorial_radius',
'doblateness', and 'dposition_angle'. Default values are set to zero.
Search done between (value - dvalue) and (value + dvalue).


Examples
--------
To fit the ellipse to the chords of occ1 Occultation object:

>>> fit_ellipse(occ1, **kwargs)

To fit the ellipse to the chords of occ1 and occ2 Occultation objects together:

>>> fit_ellipse(occ1, occ2, **kwargs)
```

(continues on next page)

(continued from previous page)

```
File:      ~/Documentos/códigos/SORA/sora/occultation/core.py
Type:      method
```

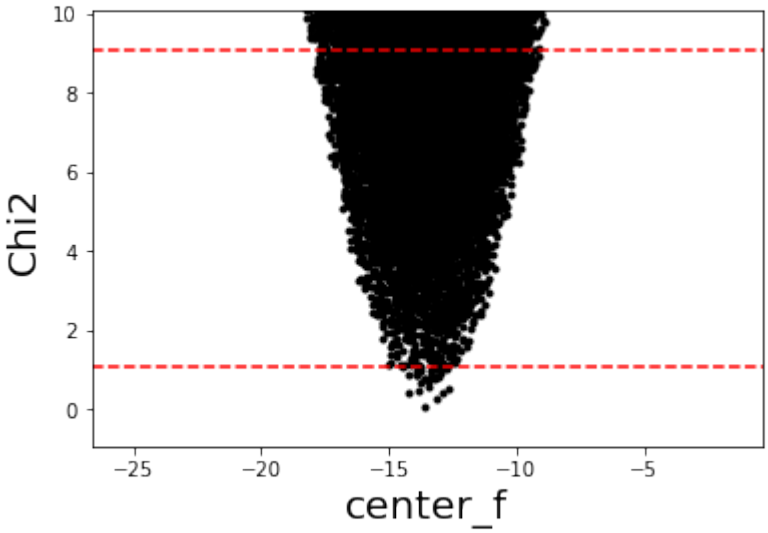
[31]:
```
ellipse_chi2  = occ.fit_ellipse(center_f=-15.046, center_g=-2.495, dcenter_f=10, dcenter_
↪g=20,
                                equatorial_radius=138, dequatorial_radius=50,␣
↪oblateness=0.093,
                                doblateness=0.20, position_angle=90, dposition_angle=90 ,
↪loop=100000,
                                dchi_min=10,number_chi=20000)
```

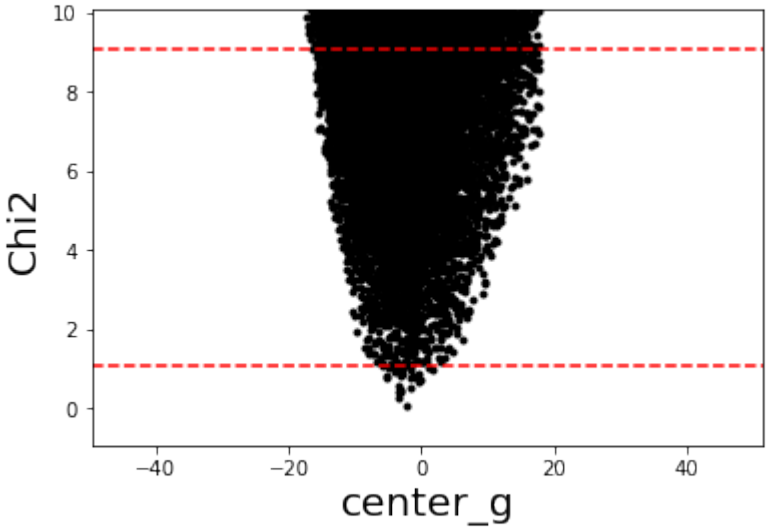
Note that here we only used 100000 loops, but it is recommended at least 10 millions for a good sample for each parameter.

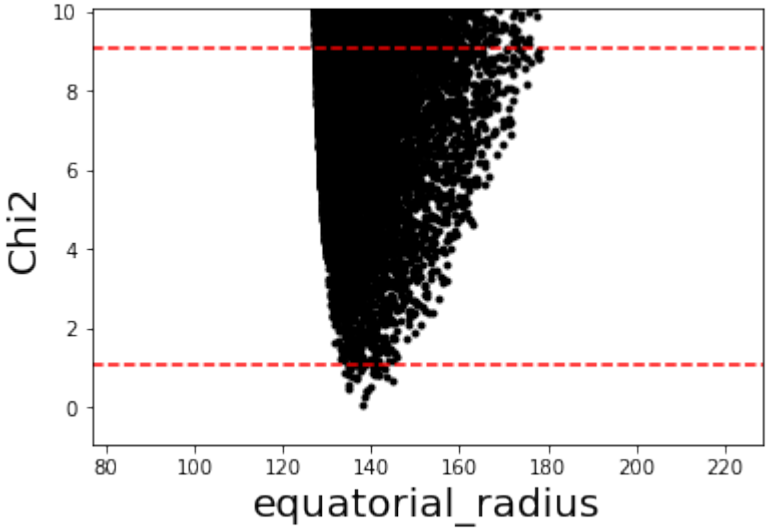
[32]:
```
print(ellipse_chi2)
```

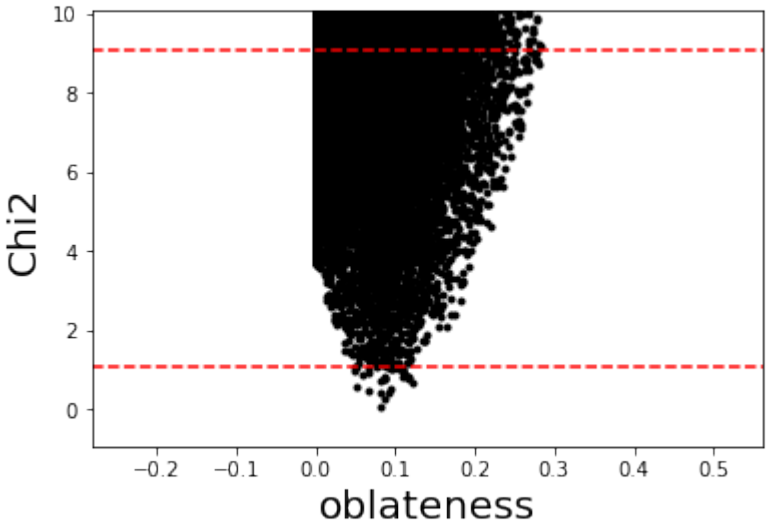
```
Minimum chi-square: 0.062
Number of fitted points: 10
Number of fitted parameters: 5
Minimum chi-square per degree of freedom: 0.012

center_f:
    1-sigma: -13.429 +/- 0.789
    3-sigma: -13.488 +/- 4.379

center_g:
    1-sigma: -1.969 +/- 3.795
    3-sigma: 0.912 +/- 16.823

equatorial_radius:
    1-sigma: 139.351 +/- 5.457
    3-sigma: 152.721 +/- 25.282

oblateness:
    1-sigma: 0.085 +/- 0.037
    3-sigma: 0.140 +/- 0.140

position_angle:
    1-sigma: 126.475 +/- 18.116
    3-sigma: 125.088 +/- 89.978
```

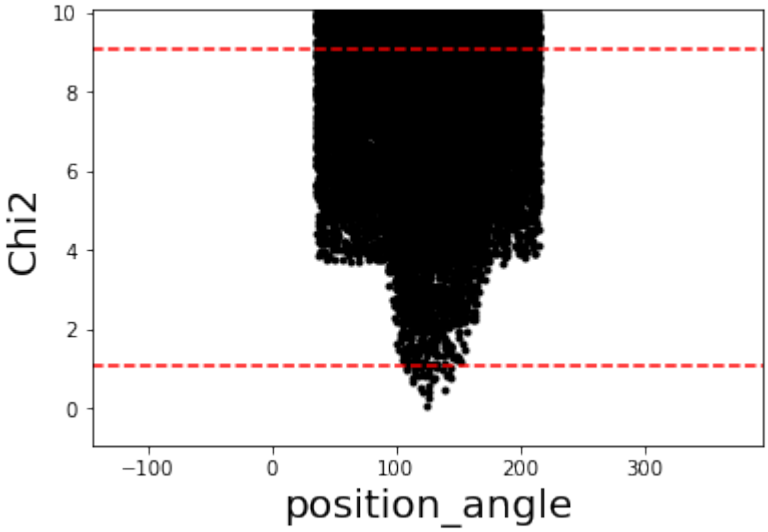
[33]:
```
ellipse_chi2.plot_chi2()
```


Chi2
center_f


Chi2
center_g

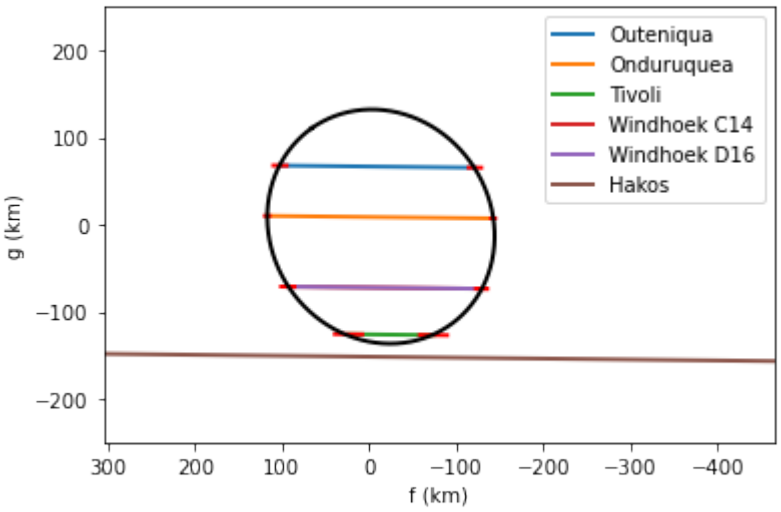

Chi2
equatorial_radius





Now, besides the chords we can plot the fitted ellipse

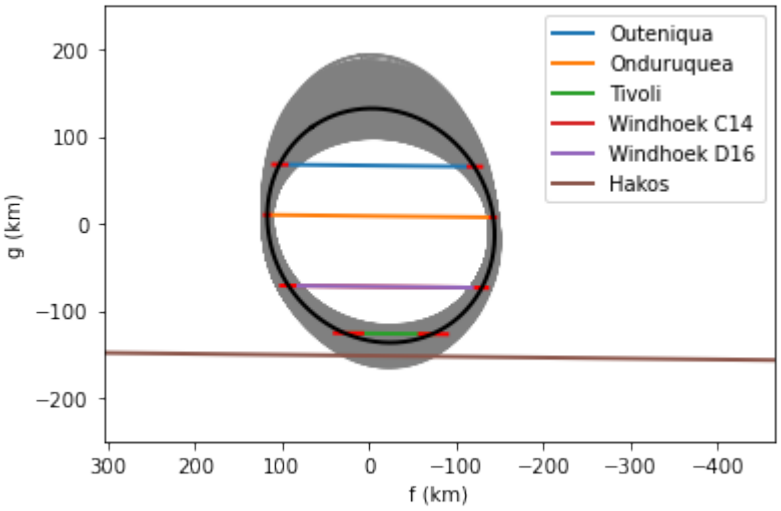
[34]:
```
from sora.extra import draw_ellipse

occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

#plotting the best fitted ellipse, in black
draw_ellipse(**ellipse_chi2.get_values())

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```



Not just the fitted ellipse, but the user can also plot all the ellipses inside an sigma region, for instance within :math:`3sigma`

This step will, possible, plot large number of ellipses, this can spend some CPU time.

[35]:
```
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

#plotting the best fitted ellipse, in black
draw_ellipse(**ellipse_chi2.get_values())

# ploting all the ellipses within 3-sigma, in gray
draw_ellipse(**ellipse_chi2.get_values(sigma=3),alpha=1.0,lw=1)

pl.legend(loc=1)
pl.xlim(+170,-330)
pl.ylim(-250,+250)
pl.show()
```



As can be seen, there are some ellipses that cross the Hakos negative chord. They are impossible solutions

as the Hakos lightcurve show no evidence of an occultation. The User can filter this solutions using the `filter_negative_chord()` function.

[36]:
```
for from sora.occultation import filter_negative_chord
```

[37]:
```
filter_negative_chord?
```

```
Signature: filter_negative_chord(chord, chisquare, step=1, sigma=0)
Docstring:
Get points for the ellipse with the given input parameters.

Parameters
----------
chord : `sora.observer.Chord`
    Chord object, must be associated to an Occultation to work.

chisquare : `sora.extra.ChiSquare`
    Resulted ChiSquare object of fit_ellipse.

sigma : `int`, `float`
    Uncertainty of the expected ellipse, in km.

step : `int`, `float`, `str`
    If a number, it corresponds to the step, in seconds, for each point of
    the chord path. The step can also be equal to ``'exposure'``. In this
    case, the chord path will consider the lightcurve individual times and
    exptime.
File:      ~/Documentos/códigos/SORA/sora/occultation/utils.py
Type:      function
```

[38]:
```
filter_chi2 = filter_negative_chord(chord=occ.chords['Hakos'], chisquare=ellipse_chi2,␣
↪step=0.5)

print(filter_chi2)
```

```
Filter chord: Hakos |...|  - 14%
```

```
IOPub message rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_msg_rate_limit`.

Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

```
Filter chord: Hakos |...|  - 37%
```

```
IOPub message rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
```

(continues on next page)

(continued from previous page)

```
`--ServerApp.iopub_msg_rate_limit`.

Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

```
Filter chord: Hakos |...|  - 60%
```

```
IOPub message rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_msg_rate_limit`.

Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

```
Filter chord: Hakos |...|  - 84%
```

```
IOPub message rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_msg_rate_limit`.

Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

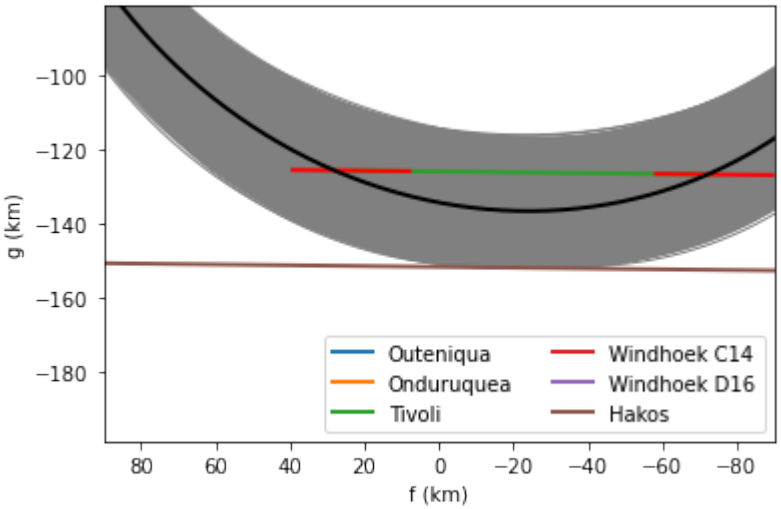
```
Filter chord: Hakos ||  - 100%
Minimum chi-square: 0.062
Number of fitted points: 10
Number of fitted parameters: 5
Minimum chi-square per degree of freedom: 0.012

center_f:
    1-sigma: -13.429 +/- 0.789
    3-sigma: -13.488 +/- 4.379

center_g:
    1-sigma: -1.969 +/- 3.795
    3-sigma: 0.912 +/- 16.823

equatorial_radius:
    1-sigma: 139.351 +/- 5.457
    3-sigma: 149.280 +/- 21.842

oblateness:
    1-sigma: 0.085 +/- 0.037
    3-sigma: 0.131 +/- 0.131
```

(continues on next page)

(continued from previous page)

```
position_angle:
    1-sigma: 126.475 +/- 18.116
    3-sigma: 125.088 +/- 89.978
```

[39]:
```
occ.chords.plot_chords()
occ.chords.plot_chords(segment='error', color='red')

#plotting the best fitted ellipse, in black
draw_ellipse(**filter_chi2.get_values())

# ploting all the ellipses within 3-sigma, in gray
draw_ellipse(**filter_chi2.get_values(sigma=3),alpha=1.0,lw=1)

pl.legend(loc=4,ncol=2)
pl.xlim(+90,-90)
pl.ylim(-180,-100)
pl.show()
```



The user can set `step = "exposure"` and consider only the region where data was acquired, allowing ellipses that crosses the negative chord during its readout time.

[40]:
```
filter_2_chi2 = filter_negative_chord(chord = occ.chords['Hakos'], chisquare = ellipse_
↪chi2, step='exposure')

print(filter_2_chi2)
```

```
Filter chord: Hakos |...|  - 14%
```

```
IOPub message rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_msg_rate_limit`.
```

(continues on next page)

(continued from previous page)

```
Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

```
Filter chord: Hakos |...|  - 38%
```

```
IOPub message rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_msg_rate_limit`.

Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

```
Filter chord: Hakos |...|  - 62%
```

```
IOPub message rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_msg_rate_limit`.

Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

```
Filter chord: Hakos |...|  - 88%
```

```
IOPub message rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_msg_rate_limit`.

Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

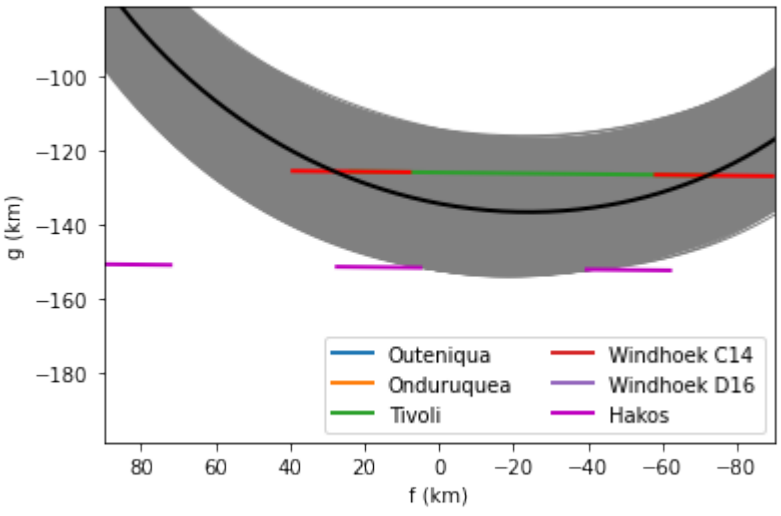
```
Filter chord: Hakos ||  - 100%
Minimum chi-square: 0.062
Number of fitted points: 10
Number of fitted parameters: 5
Minimum chi-square per degree of freedom: 0.012

center_f:
    1-sigma: -13.429 +/- 0.789
    3-sigma: -13.488 +/- 4.379

center_g:
    1-sigma: -1.969 +/- 3.795
```

(continues on next page)

(continued from previous page)

```
    3-sigma: 0.912 +/- 16.823

equatorial_radius:
    1-sigma: 139.351 +/- 5.457
    3-sigma: 149.280 +/- 21.842

oblateness:
    1-sigma: 0.085 +/- 0.037
    3-sigma: 0.131 +/- 0.131

position_angle:
    1-sigma: 126.475 +/- 18.116
    3-sigma: 125.088 +/- 89.978
```

[41]:
```
occ.chords.plot_chords(segment='positive')
occ.chords.plot_chords(segment='error', color='red')

chord2.plot_chord(segment='negative', linestyle='exposure', color='m', label='Hakos')

#plotting the best fitted ellipse, in black
draw_ellipse(**filter_2_chi2.get_values())

# ploting all the ellipses within 3-sigma, in gray
draw_ellipse(**filter_2_chi2.get_values(sigma=3),alpha=1.0,lw=1)

pl.legend(loc=4,ncol=2)
pl.xlim(+90,-90)
pl.ylim(-180,-100)
pl.show()
```



The radial velocity issue

During the `LightCurve` fitting, the user has to add the velocity for that light curve. In the initial process, only the geocentric velocity was determined, and only after the fit, a radial velocity can be correctly calculated.

Usually, the light curves' features are dominated by the exposure time. However, in the cases that the fresnel diffraction plays a significant role in it, we suggest that the `LightCurve` fitting procedure should be redone with the correct velocities.

[42]:
```
occ.check_velocities()
```

```
Outeniqua - Velocity used: 22.004
    Immersion Radial Velocity: 17.800
    Emersion Radial Velocity: 18.284
Onduruquea - Velocity used: 22.004
    Immersion Radial Velocity: 22.252
    Emersion Radial Velocity: 22.246
Tivoli - Velocity used: 22.004
    Immersion Radial Velocity: 8.626
    Emersion Radial Velocity: 5.195
Windhoek C14 - Velocity used: 22.004
    Immersion Radial Velocity: 18.170
    Emersion Radial Velocity: 17.470
Windhoek D16 - Velocity used: 22.004
    Immersion Radial Velocity: 18.107
    Emersion Radial Velocity: 17.526
```

9.6.5 4. Viewing and saving the results

In the end, we can see the results using Python dictionaries. The `Occultation.fitted_params` will have fitted parameters and their 1σ uncertainties. The `Occultation.chi2_params` will have some information about the fit and its quality.

[43]:
```
occ.fitted_params
```

[43]:
```
{'equatorial_radius': [139.3506109843088, 5.4567114516244],
 'center_f': [-13.428622523637122, 0.7891892760690086],
 'center_g': [-1.9686948831613296, 3.795102365340013],
 'oblateness': [0.08488597520674623, 0.03706668452200804],
 'position_angle': [126.47517825871333, 18.116387367826775]}
```

[44]:
```
occ.chi2_params
```

[44]:
```
{'chord_name': ['Outeniqua_immersion',
  'Outeniqua_emersion',
  'Onduruquea_immersion',
  'Onduruquea_emersion',
  'Tivoli_immersion',
  'Tivoli_emersion',
  'Windhoek C14_immersion',
  'Windhoek C14_emersion',
  'Windhoek D16_immersion',
  'Windhoek D16_emersion'],
 'radial_dispersion': array([-0.14982613, -2.3388093 , -1.21884535, -1.3706693 , -0.
↪17265982,
        -2.22234731, -0.58491025, -1.06365652, -1.66000562, -0.34293474]),
 'position_angle': array([301.88964659,  58.90450147, 274.03961814,  84.75566445,
        205.85090919, 163.36646323, 238.44925955, 123.18339268,
```

(continues on next page)

(continued from previous page)

```
       238.18760663, 122.96493764]),
 'radial_error': array([ 7.15442539,  7.60165495,  2.23532561,  2.45888165, 15.65002209,
       15.65007494,  5.36529313,  5.81244286,  6.25950441,  7.60088706]),
 'chi2_min': 0.06220013599688508,
 'nparam': 5,
 'npts': 10}
```

The astrometrical positions obtained can be seen using `Occultation.new_astrometric_position()`

[45]:
```
occ.new_astrometric_position()
```

```
Ephemeris offset (km): X = -13.4 km +/- 0.8 km; Y = -2.0 km +/- 3.8 km
Ephemeris offset (mas): da_cos_dec = -1.263 +/- 0.074; d_dec = -0.185 +/- 0.357

Astrometric object position at time 2017-06-22 21:18:48.200 for reference 'geocenter'
RA = 18 55 15.6523925 +/- 0.081 mas; DEC = -31 31 21.622083 +/- 0.359 mas
```

If the user wants the position at a specific time

[46]:
```
occ.new_astrometric_position('2017-06-22 21:21:00.000')
```

```
Ephemeris offset (km): X = -13.4 km +/- 0.8 km; Y = -2.0 km +/- 3.8 km
Ephemeris offset (mas): da_cos_dec = -1.263 +/- 0.074; d_dec = -0.185 +/- 0.357

Astrometric object position at time 2017-06-22 21:21:00.000 for reference 'geocenter'
RA = 18 55 15.6310596 +/- 0.081 mas; DEC = -31 31 21.623477 +/- 0.359 mas
```

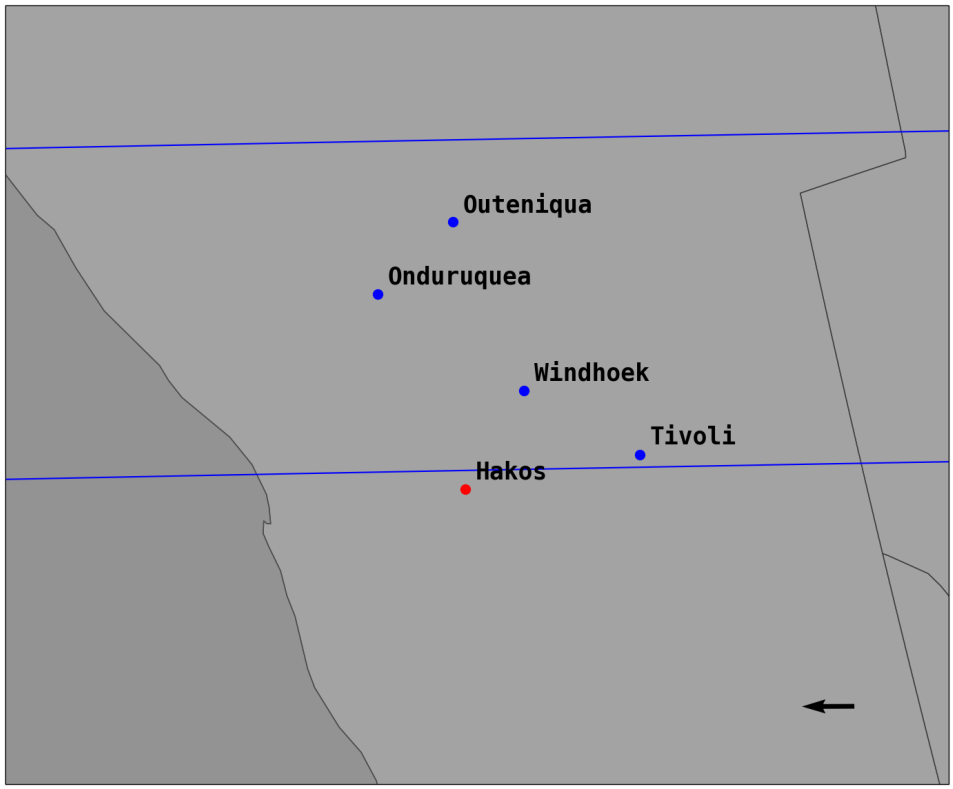
Also a 'post-fit' occultation map can be created using `Occultation.plot_occ_map()`

The function that plots the map contains a large number of *kwargs*. They can be used to completely control the map and its tutorial can be found at SORA documentation here.

[47]:
```
occ.plot_occ_map(centermap_delta=[-3500,+400],zoom=20,nameimg='output/map_posfit')
```

```
Projected shadow radius = 135.2 km
output/map_posfit.png generated
```

```
Object        Diam   Tmax   dots <> ra_offset_dec
Chariklo    270 km   12.3s  60 s <>  -1.4   -0.1
```




```
 year-m-d    h:m:s UT     ra__dec__J2000__candidate    C/A    P/A    vel   Delta   G*  long
2017-06-22 21:18:48.917  18 55 15.6525 -31 31 21.671  0.049 359.72 -22.00  14.66  14.3   53
```

All this information can also be seen using `print(Occultation)`

[48]:
```
print(occ)
```

```
Stellar occultation of star GaiaDR3 6760223758801661440 by 10199 Chariklo (1997 CU26).

Geocentric Closest Approach: 0.049 arcsec
Instant of CA: 2017-06-22 21:18:48.200
Position Angle: 359.72 deg
Geocentric shadow velocity: -22.00 km / s
Sun-Geocenter-Target angle:  166.42 deg
Moon-Geocenter-Target angle: 149.11 deg


5 positive observations
1 negative observations

################################################################################
                                    STAR
################################################################################
GaiaDR3 star Source ID: 6760223758801661440
```

(continues on next page)

(continued from previous page)

```
ICRS star coordinate at J2016.0:
RA=18h55m15.65210s +/- 0.0197 mas, DEC=-31d31m21.6676s +/- 0.0180 mas
pmRA=3.556 +/- 0.025 mas/yr, pmDEC=-2.050 +/- 0.020 mas/yr
GaiaDR3 Proper motion corrected as suggested by Cantat-Gaudin & Brandt (2021)
Plx=0.2121 +/- 0.0228 mas, Rad. Vel.=-40.49 +/- 3.73 km/s

Magnitudes: G: 14.224, B: 14.320, V: 13.530, R: 14.180, J: 12.395, H: 11.781,
            K: 11.627

Apparent diameter from Kervella et. al (2004):
    V: 0.0216 mas, B: 0.0216 mas
Apparent diameter from van Belle (1999):
    sg: B: 0.0238 mas, V: 0.0244 mas
    ms: B: 0.0261 mas, V: 0.0198 mas
    vs: B: 0.0350 mas, V: 0.0315 mas

Geocentric star coordinate at occultation Epoch (2017-06-22 21:18:48.200):
RA=18h55m15.65251s +/- 0.0327 mas, DEC=-31d31m21.6706s +/- 0.0341 mas

################################################################################
                          10199 Chariklo (1997 CU26)
################################################################################
Object Orbital Class: Centaur
Spectral Type:
    SMASS: D  [Reference: EAR-A-5-DDR-TAXONOMY-V4.0]
       Relatively featureless spectrum with very steep red slope.
Discovered 1997-Feb-15 by Spacewatch at Kitt Peak

Physical parameters:
Diameter:
    302 +/- 30 km
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),
Rotation:
    7.004 +/- 0 h
    Reference: LCDB (Rev. 2021-June); Warner et al., 2009, [Result based on less than␣
↪full coverage, so that the period may be wrong by 30 percent or so.]  REFERENCE LIST:
↪[Fornasier, S.; Lazzaro, D.; Alvarez-Candal, A.; Snodgrass, C.; et al. (2014) Astron.␣
↪Astrophys. 568, L11.], [Leiva, R.; Sicardy, B.; Camargo, J.I.B.; Desmars, J.; et al.␣
↪(2017) Astron. J. 154, A159.]
Absolute Magnitude:
    6.54 +/- 0 mag
    Reference: JPL Horizons,
Phase Slope:
    0.15 +/- 0
    Reference: JPL Horizons,
Albedo:
    0.045 +/- 0.01
    Reference: Earth, Moon, and Planets, v. 89, Issue 1, p. 117-134 (2002),

Ellipsoid: 151.0 x 151.0 x 151.0

----------- Ephemeris -----------
```

(continues on next page)

(continued from previous page)

```
EphemKernel: CHARIKLO/DE438_SMALL (SPKID=2010199)
Ephem Error: RA*cosDEC: 0.000 arcsec; DEC: 0.000 arcsec
Offset applied: RA*cosDEC: 0.0000 arcsec; DEC: 0.0000 arcsec


################################################################################
                              POSITIVE OBSERVATIONS
################################################################################

--------------------------------------------------------------------------------
Site: Outeniqua
Geodetic coordinates: Lon: 16d49m17.71s, Lat: -21d17m58.17s, height: 1.416 km
Target altitude: 56.6 deg
Target azimuth:  115.3 deg

Light curve name: Outeniqua lc
Initial time: 2017-06-22 21:20:00.056 UTC
End time:     2017-06-22 21:29:59.963 UTC
Duration:     9.998 minutes
Time offset:  -0.150 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:20.179 UTC +/- 0.320 seconds
Emersion time:  2017-06-22 21:21:30.193 UTC +/- 0.340 seconds


--------------------------------------------------------------------------------
Site: Onduruquea
Geodetic coordinates: Lon: 15d59m33.75s, Lat: -21d36m26.04s, height: 1.220 km
Target altitude: 56.1 deg
Target azimuth:  114.7 deg

Light curve name: Onduruquea lc
Initial time: 2017-06-22 21:11:52.175 UTC
End time:     2017-06-22 21:25:13.389 UTC
Duration:     13.354 minutes
Time offset:  -0.190 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
```

(continues on next page)

(continued from previous page)

```
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:22.023 UTC +/- 0.100 seconds
Emersion time:  2017-06-22 21:21:33.634 UTC +/- 0.110 seconds


--------------------------------------------------------------------------------
Site: Tivoli
Geodetic coordinates: Lon: 18d01m01.24s, Lat: -23d27m40.19s, height: 1.344 km
Target altitude: 58.5 deg
Target azimuth:  112.4 deg

Light curve name: Tivoli lc
Initial time: 2017-06-22 21:16:00.094 UTC
End time:     2017-06-22 21:28:00.018 UTC
Duration:     11.999 minutes
Time offset:  -0.150 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:15.478 UTC +/- 0.700 seconds
Emersion time:  2017-06-22 21:21:19.838 UTC +/- 0.700 seconds


--------------------------------------------------------------------------------
Site: Windhoek
Geodetic coordinates: Lon: 17d06m31.9s, Lat: -22d41m55.16s, height: 1.902 km
Target altitude: 57.4 deg
Target azimuth:  113.4 deg

Light curve name: Windhoek C14 lc
Initial time: 2017-06-22 21:12:48.250 UTC
End time:     2017-06-22 21:32:47.963 UTC
Duration:     19.995 minutes
Time offset:  -0.375 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
```

(continues on next page)

(continued from previous page)

```
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:17.234 UTC +/- 0.240 seconds
Emersion time:  2017-06-22 21:21:27.189 UTC +/- 0.260 seconds


--------------------------------------------------------------------------------
Site: Windhoek
Geodetic coordinates: Lon: 17d06m31.9s, Lat: -22d41m55.16s, height: 1.902 km
Target altitude: 57.4 deg
Target azimuth:  113.4 deg

Light curve name: Windhoek D16 lc
Initial time: 2017-06-22 21:20:01.884 UTC
End time:     2017-06-22 21:22:21.894 UTC
Duration:     2.333 minutes
Time offset:  0.000 seconds

Object LightCurve was not instantiated with time and flux.

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion time: 2017-06-22 21:21:17.288 UTC +/- 0.280 seconds
Emersion time:  2017-06-22 21:21:27.228 UTC +/- 0.340 seconds


################################################################################
                              NEGATIVE OBSERVATIONS
################################################################################

--------------------------------------------------------------------------------
Site: Hakos
Geodetic coordinates: Lon: 16d21m41.32s, Lat: -23d14m11.04s, height: 1.843 km
Target altitude: 56.8 deg
Target azimuth:  112.5 deg

Light curve name: Hakos lc
Initial time: 2017-06-22 21:10:19.961 UTC
End time:     2017-06-22 21:30:16.955 UTC
Duration:     19.950 minutes
Time offset:  0.000 seconds
```

(continues on next page)

(continued from previous page)

```
Exposure time:    1.0000 seconds
Cycle time:       2.9998 seconds
Num. data points: 400

Bandpass:             0.700 +/- 0.300 microns
Object Distance:      14.66 AU
Used shadow velocity: 22.004 km/s
Fresnel scale:        0.040 seconds or 0.87 km
Stellar size effect:  0.010 seconds or 0.23 km

Object LightCurve model was not fitted.

Immersion and emersion times were not fitted or instantiated.


################################################################################
                                    RESULTS
################################################################################

Fitted Ellipse:
equatorial_radius: 139.351 +/- 5.457
center_f: -13.429 +/- 0.789
center_g: -1.969 +/- 3.795
oblateness: 0.085 +/- 0.037
position_angle: 126.475 +/- 18.116
polar_radius: 127.522 km
equivalent_radius: 133.305 km
geometric albedo (V): 0.061 (6.1%)

Minimum chi-square: 0.062
Number of fitted points: 10
Number of fitted parameters: 5
Minimum chi-square per degree of freedom: 0.012
Radial dispersion: -1.112 +/- 0.803 km
Radial error:      7.579 +/- 4.655 km

Ephemeris offset (km): X = -13.4 km +/- 0.8 km; Y = -2.0 km +/- 3.8 km
Ephemeris offset (mas): da_cos_dec = -1.263 +/- 0.074; d_dec = -0.185 +/- 0.357

Astrometric object position at time 2017-06-22 21:18:48.200 for reference 'geocenter'
RA = 18 55 15.6523925 +/- 0.081 mas; DEC = -31 31 21.622083 +/- 0.359 mas
```

And this can be saved to an ASCII file using `Occultation.to_log`

[49]:
```
occ.to_log('output/Test_occ.log')

!ls output/*.log
```

```
output/LC_test_1a.log  output/Test_occ.log
```

This Jupyter-Notebook was designed as a tutorial for how to work with the `Occultation` **Class. More information about the other classes, please refer to their specif Jupyter-Notebook. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo**

Carlos Boufleur.

The End

9.7 ChiSquare Class

The `ChiSquare` Class within SORA was created to control the χ^2 minimization tests and obtain the best fit values and uncertainties. Here we have some useful tasks that allows the user to get the best value and uncertainty within a sigma uncertainty value, plot and save the values. The documentation here contains the details about every step.

This Jupyter-Notebook was designed as a tutorial for how to work with the `ChiSquare` Object Class. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The `ChiSquare` Docstring was designed to help the users. Also, each function has its Docstring containing its main purpose and the needed parameters (physical description and formats). Please, do not hesitate to use it.

9.7.1 0. Index

1. *Instantiating the ChiSquare Object*
2. *Viewing and saving the results*
3. *Saving the results and plots*

[1]:
```
## SORA package
from sora.extra import ChiSquare

## Usual packages
import numpy as np
import matplotlib.pyplot as pl
```

```
SORA version: 0.3
```

9.7.2 1. Instantiating a ChiSquare Object

The `ChiSquare` Class was not created to be instantiated by the user. Its main function is to control the fitting process and it will be resulting of the fitting tasks, such as `LightCurve.occ_lcfit()` and `Occultation.fit_ellipse()`. This Object contains a python dictionary with the χ^2 values and the associated tested parameters.

[2]:
```
ChiSquare?
```

```
Init signature: ChiSquare(chi2, npts, **kwargs)
Docstring:
Stores the arrays for all inputs and given chi-square.

Parameters
----------
chi2 : `array`
    Array with all the chi-square values.

npts : `int`
    Number of points used in the fit.
```

(continues on next page)

(continued from previous page)

```
**kwargs
    Any other given input must be an array with the same size as chi2. The
    keyword `name` will be associated as the variable `name` of the given data.

Example
-------
>>> chisquare = ChiSquare(chi2, immersion=t1, emersion=t2)

``t1`` and ``t2`` must be an array with the same size as chi2.

The data can be accessed as

>>> chisquare.data['immersion']
File:           ~/Documentos/códigos/SORA/sora/extra/chisquare.py
Type:           type
Subclasses:
```

[3]:
```
chi2, t_immersion, t_emersion = np.loadtxt('input/ascii/example_chi2.dat',unpack=True)

chisquare = ChiSquare(chi2=chi2, npts=496, t_immersion=t_immersion, t_emersion=t_
↪emersion)
```

9.7.3 2. Viewing the results

First, the value that minimizes the χ^2 can be obtained using the `ChiSquare.get_values()`.

[4]:
```
chisquare.get_values()
```

[4]:
```
{'t_immersion': 76880.33436, 't_emersion': 76890.34416}
```

All the values within a sigma limit can be found in a similar manner

[5]:
```
chisquare.get_values(sigma=3)
```

[5]:
```
{'t_immersion': array([76880.37133, 76880.28921, 76880.24912, ..., 76880.29041,
        76880.39328, 76880.43463]),
 't_emersion': array([76890.26501, 76890.25714, 76890.37106, ..., 76890.42023,
        76890.4073 , 76890.36891])}
```

The best-fitted value and its uncertainties can be obtained using `ChiSquare.get_nsigma()`

[6]:
```
chisquare.get_nsigma()
```

[6]:
```
{'chi2_min': 489.84066,
 'sigma': 1,
 'n_points': 137,
 't_immersion': [76880.32901, 0.028059999996912666],
 't_emersion': [76890.34104, 0.03118999999423977]}
```

[7]:
```
chisquare.get_nsigma(sigma=3)
```

[7]:
```
{'chi2_min': 489.84066,
 'sigma': 3,
 'n_points': 1719,
 't_immersion': [76880.36540000001, 0.13475000000471482],
 't_emersion': [76890.34015999999, 0.10841999999684049]}
```

Also, the number of fitted points (N) and the number of fitted parameters (P) can be obtained using `ChiSquare.npts` **and** `ChiSquare.nparam`

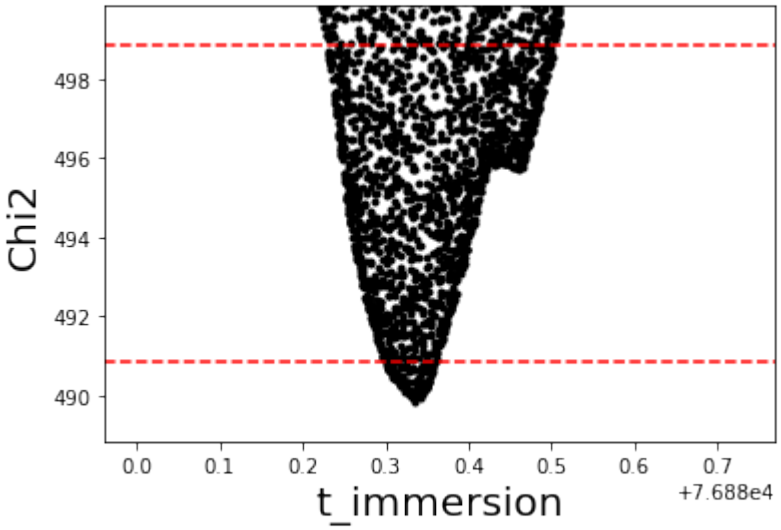
[8]:
```
print('Number of fitted points: {}'.format(chisquare.npts))
print('Number of fitted parameters: {}'.format(chisquare.nparam))
```

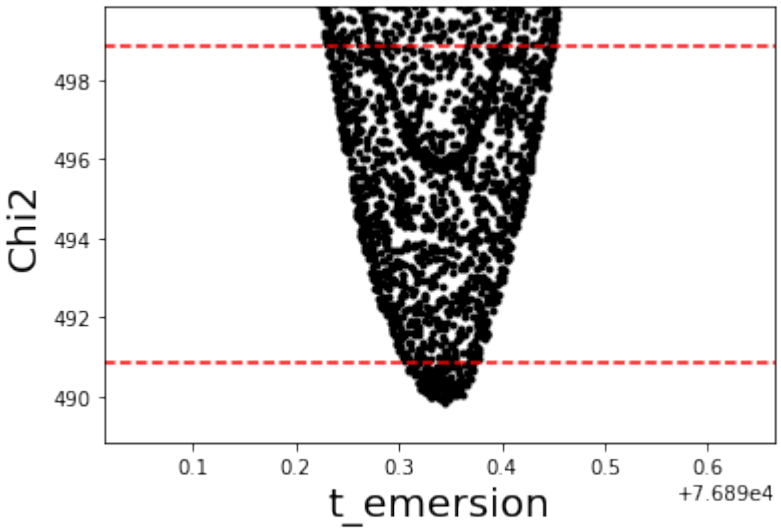
```
Number of fitted points: 496
Number of fitted parameters: 2
```

9.7.4 3. Saving the results and plots

The plot with the χ^2 by fitted parameter can be seen using `ChiSquare.plot_chi2()`

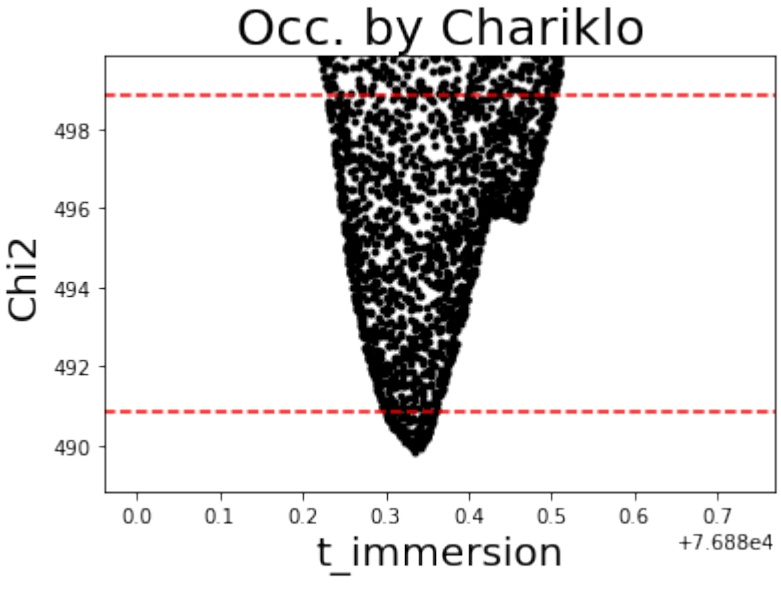
[9]:
```
chisquare.plot_chi2()
```





If the user choose to plot only one, it is possible to have complete control over it using *matplotlib* functions

[10]:
```
chisquare.plot_chi2('t_immersion')

pl.title('Occ. by Chariklo',fontsize=25)
pl.show()
```



The values within the `ChiSquare` object can be saved as an ASCII file using `ChiSquare.to_file()`

[11]:
```
chisquare.to_file('output/example_chi2.dat')
```

Two `ChiSquare` objects can be easily combined by adding them

[12]:
```
chisquare_2 = chisquare + chisquare

print('Single ChiSquare:  ',len(chisquare.data['chi2']))
```

(continues on next page)

(continued from previous page)

```
print('Combined ChiSquare:',len(chisquare_2.data['chi2']))
```

```
Single ChiSquare:   10000
Combined ChiSquare: 20000
```

This Jupyter-Notebook was designed as a tutorial for how to work with the `ChiSquare` Class. More information about the other classes, please refer to their specif Jupyter-Notebook. Any further question, please contact the core team: Altair Ramos Gomes Júnior, Bruno Eduardo Morgado, Gustavo Benedetti Rossi, and Rodrigo Carlos Boufleur.

The End

CHAPTER

TEN

KNOWN ISSUES

10.1 Progress Bar Output

- Jupyter Notebooks running on Windows OS will experience missing progress bar outputs when setting *"threads"* with a value >1 in the *LightCurve.occ_lcfit()* and the *Occultation.fit_ellipse()* functions. The issue arises from how the multiprocessing python module was implemented in Windows OS and how the SORA multithreading progress bars piping output was designed. Nonetheless, the progress bar output will still be printed in the execution terminal. [#79]

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

U

V