OCFit - manual

Pavol Gajdoš

2019

version 0.1.2

Contents

1 Introduction

Package OCFit includes 4 classes for analysis and fitting of O-C diagrams. This manual is brief summary of available function with their basic description.

In a case of using this package for scientific purposes, please, cite our paper Gajdoš & Parimucha (2018) where you can also find more detail description about fitting functions and used models.

2 Necessary packages

This software works in Python 2.7 with these packages:

- numpy - tested on version 1.10.2
- matplotlib - tested on version 1.5.0
- PyAstronomy - tested on version 0.9.0
- pymc - tested on version 2.3.6
- ga.py - own file which is necessary for fitting using genetic algorithms
- info_ga.py - own file for an analysis of fitting using genetic algorithms
- info_mc.py - own file for an analysis of fitting using Monte Carlo method

3 Installation

Installation is possible from source code or using build installation binary file (only for OS Windows). The following procedure is only for installation from the source code. Extract files and go to new-created folder. Running script setup.py the installation will be done:

```
python setup.py install
```

4 FitLinear and FitQuad

Classes are used for fitting a linear a quadratic trend on O-C diagram caused by mass transfer. It is also possible to calculate O-C values from observed times of minima ($O \equiv t_O$). The class FitLinear could also be used for removing linear trend in the data and then fit only quadratic trend, see Example.

Functions and parameters of the both classes are nearly identical. In the case, that some function is only in one function, it is highlighted.

4.1 Initialization

OCFit.**FitLinear**(t, t0, P, oc = [ ], err = [ ])
OCFit.**FitQuad**(t, t0, P, oc = [ ], err = [ ])

For initialization these classes, it is necessary to give observed times of minima (O) and linear ephemeris of minima, i. e. time of reference minimum and period. If the values of O-C are already calculated, they could be given as the next parameter. If not, they will be automatically calculated from given times. These values could be immediately used in other calculations, they are stored in variable oc. It is also possible to give errors or uncertainties of times of minima (and also O-Cs). Eventually, if there are given only weights of individual points, it is possible to enter error as an invert value of weight.

Parameters:

t: numpy.array or list

Observed times of minima (O).

t0: float

Time of reference minimum.

P: float

Period of the system.

oc: numpy.array or list, optional

Calculated values of O-Cs in days.

err: numpy.array or list, optional

Errors of O-C values or times O.

4.2 Available functions

4.2.1 FitLinear

OCFit.FitLinear.**FitLinear**()

The basic function for fitting linear trend in O-C diagram. **This function is only available in a class FitLinear.** New values of linear ephemeris are written to variable params; their uncertainties are in variable params_err. The new times (C) are also calculated stored in variable tC and new values of O-C stored in variable new_oc.

Output: numpy.array

New values of O-C calculated on the basis of new linear ephemeris in days.

4.2.2 FitQuad

OCFit.FitQuad.**FitQuad**()

The basic function for fitting quadratic trend in O-C diagram. **This function is only available in a class FitQuad.** New values of Q and linear ephemeris are written to variable params; their uncertainties are in variable params_err. The new times (C) are also calculated stored in variable tC and new values of O-C stored in variable new_oc.

Output: numpy.array

New values of O-C for calculated Q and linear ephemeris in days.

4.2.3 FitRobust

OCFit.FitLinear.**FitRobust**(n_iter = 10)
OCFit.FitQuad.**FitRobust**(n_iter = 10)

The function for fitting linear / quadratic trend in O-C diagrams using robust regression. It is suitable to use this function in cases if there are a lot of outlier points in the data. New values of fitted parameters are written to variable params; their uncertainties are in variable params_err. The new times (C) are also calculated stored in variable tC and new values of O-C stored in variable new_oc.

Parameters:

n_iter: integer, optional

Number of iterations.

Output: numpy.array

New values of O-C.

4.2.4 FitMCMC

OCFit.FitLinear.**FitMCMC**(n_iter , limits, steps, fit_params = ['P', 't0'], burn = 0, binn = 1, visible = True, db = '')
OCFit.FitQuad.**FitMCMC**(n_iter , limits, steps, fit_params = ['Q', 'P', 't0'], burn = 0, binn = 1, visible = True, db = '')

The function for fitting linear / quadratic trend in O-C diagram using Monte Carlo (MC) method. Running of this function is possible only if the initial values of fitted parameters were calculated. They could be obtained from fitting using function FitLinear / FitQuad or FitRobust. New values of fitted parameters are written to variable `params`; their uncertainties are in variable `params_err`. The new times (C) are also calculated stored in variable `tC` and new values of O-C stored in variable `new_oc`.

More detail description of Monte Carlo method is in a part about the class OCFit (subsection 5.5.20).

Parameters:

n_iter: float

Number of MC steps.

limits: dictionary

Limits for fitted parameters.

steps: dictionary

The steps for fitted parameters.

fit_params: list

List of fitted parameters.

burn: float, optional

Number of MC steps which will be reject from calculation.

binn: float, optional

The size of one block for binning.

db: string, optional

Name of file for saving MC sampling.

visible: boolean, optional

Display progress of fitting and summary of results from `pymc`.

Output: numpy.array

New values of O-C.

4.2.5 InfoMCMC

OCFit.FitLinear.**InfoMCMC**(db, eps = False, geweke = False)
OCFit.FitQuad.**InfoMCMC**(db, eps = False, geweke = False)

Function for detail analysis of MC sampling obtained by FitMCMC. It will be generated a plot with trace of fitting and common plot with trace and histogram for each of fitted parameters. An also, the file with brief informations about sampling, obtained values and errors of the parameter will be created. It is also possible to plot figures for Geweke analysis (details in Geweke (1992)).

Few common figures for all fitted parameters will be generated, too. There are histograms, deviances, correlation plots and 2D histograms displaying the confidence regions. For 2D histograms, the function ConfInt is used, everyone else are generated by the class TraceAnalysis from PyAstronomy.

The last output is table with correlation coefficients between fitted parameters.

Parameters:

db: string

Name of file with saved MC sampling.

eps: boolean, optional

Save figures also in eps format.

geweke: boolean, optional

Geweke analysis.

4.2.6 Epoch

OCFit.FitLinear.**Epoch**()
OCFit.FitQuad.**Epoch**()

Calculation of epoch of all observed minima. This function is called during the initialization of the class. Obtained epochs are saved to variable epoch.

Output: numpy.array

Calculated epochs.

4.2.7 PhaseCurve

OCFit.FitLinear.**PhaseCurve**(P, t0, plot = False)
OCFit.FitLinear.**PhaseCurve**(P, t0, plot = False)

Calculation of phase curve on the basis of given linear ephemeris. The final phase curve could be display on a figure.

Parameters:

P: float

Period.

t0: float

Time of reference minimum.

plot: boolean, optional

Plot also a figure or only calculated phase curve.

Output: numpy.array

Phases and values of O-C.

4.2.8 CalcErr

OCFit.FitLinear.**CalcErr**()
OCFit.FitQuad.**CalcErr**()

Calculation of errors of input data on a basis of values of fitted parameters. This function could not be used before fitting the data. It could be done using function FitLinear / FitQuad or FitRobust. For all point, the same error will be assigned. The size of error will be set that the value of reduced squared error χ_r^2 will be equal 1.

Function is good to use if there is only linear / quadratic trend in the data. Subsequently, during the repeated fitting better values of errors of parameters are obtained. It is necessary to use this function before fitting using MC method by function FitMCMC if the errors of input data were not given during initialization.

Output: numpy.array

Calculated errors.

4.2.9 CorrectErr

OCFit.FitLinear.**CorrectErr**()
OCFit.FitQuad.**CorrectErr**()

Re-normalization of given errors of input data on a basis of values of fitted parameters. This function could not be used before fitting the data. It could be done using function FitLinear / FitQuad or FitRobust. The size of error will be set that the value of reduced squared error χ_r^2 will be equal 1. For plotting the figure, original values of errors are used.

Function is good to use if there is only linear / quadratic trend in the data. Subsequently, during the repeated fitting better values of errors of parameters are obtained. It is useful to use this function before fitting using MC method by function FitMCMC if the errors of input data are wrong calculated (under- or over-estimated).

Output: numpy.array

Calculated errors.

4.2.10 AddWeight

OCFit.FitLinear.**AddWeight**(w)
OCFit.FitQuad.**AddWeight**(w)

Adding the weights and normalization of errors on a basis of values of fitted parameters. This function could not be used before fitting the data. It could be done using function FitLinear / FitQuad or FitRobust. The size of error will be set that the value of reduced squared error χ_r^2 will be equal 1.

Function is good to use if there is only linear / quadratic trend in the data. Subsequently, during the repeated fitting better values of errors of parameters are obtained. It is necessary to use this function before fitting using MC method by function FitMCMC.

Parameters:

w: numpy.array

Weights of points.

Output: numpy.array

Calculated errors.

4.2.11 Summary

OCFit.FitLinear.**Sumarry**(name = '')
OCFit.FitQuad.**Sumarry**(name = '')

Summary of values and errors of fitted parameters. The output could be saved to file or written on screen.

Parameters:

name: string, optional

Name of output file.

4.2.12 Plot

OCFit.FitLinear.**Plot**(name = '', no_plot = 0, no_plot_err = 0, eps = False, oc_min = True, time_type = 'JD', offset = 2400000, trans = True, title = '', epoch = False, min_type = False, weight = [ ], trans_weight = False, bw = False, double_ax = False)
OCFit.FitQuad.**Plot**(name = '', no_plot = 0, no_plot_err = 0, eps = False, oc_min = True, time_type = 'JD', offset = 2400000, trans = True, title = '', epoch = False, min_type = False, weight = [ ], trans_weight = False, bw = False, double_ax = False, fig_size = None)

Plotting original values of O-C and linear / quadratic fit. It is possible to save figure to png file (eventually also to eps) or only display it on the screen. Outlier points or points with big error could not be plotted. Values of O-C could be displayed in minutes or in days. It is possible to set different type of Julian date displayed on x-label. Date could be transformed by subtraction of given offset. If the date was already transformed before, it is necessary to disable transformation of x values (`trans = False`) but the value of used offset is suitable to set due to the label on x axis. It is also possible to use epochs on x axis or also have two axis with Julian dates and epochs together. In the case, if the values of O-C of primary and also secondary minima are available, it is possible to distinguished it using the parameter `min_type = True`. Primary minima are displayed be full circle and the secondary by empty circle. If the weights of points are available, they could be shown on the figures by entering them to `weight`. Weights are divided into these categories: 0-3, 3-5, 5-8, 8-10. In the case, that the weights are not in this interval, it is necessary to transform them by setting `trans_weight = True`. If the errors of data points were setted up during initialization of the class and now the weights are given to `weight`, there will be plotted figure on according to given weights without errorbars.

Parameters:

name: string, optional

Name of file (without extension) where the figure will be saved.

no_plot: integer, optional

Number of outlier points which will not be plotted.

no_plot_err: integer, optional

Number of points with the biggest errors which will not be plotted.

eps: boolean, optional

Save figure also to `eps` file. (It is necessary to set the name to variable `name`!)

oc_min: boolean, optional

Values of O-C display in minutes, not in days.

time_type: string, optional

Type of Julian date (JD, HJD, BJD).

offset: float, optional

Value of offset of dates due to shortening values on x axis.

trans: boolean, optional

Transform x values by the offset.

title: string, optional

Title of the figure.

epoch: boolean, optional

Axis x in epochs.

min_type: boolean, optional

Distinguishing primary and secondary minima.

weight: numpy.array, optional

Weights of points.

trans_weight: boolean, optional

Transformation of weights to the range 0 – 10.

bw: boolean, optional

Black and white figure.

double_ax: boolean, optional

Double x axis - dates + epochs.

fig_size: tuple, optional

Custom figure size - e.g. (12,6).

4.2.13 PlotRes

OCFit.FitLinear.**PlotRes**(name = '', no_plot = 0, no_plot_err = 0, eps = False, oc_min = True, time_type = 'JD', offset = 2400000, trans = True, title = '', epoch = False, min_type = False, weight = [ ], trans_weight = False, bw = False, double_ax = False)
OCFit.FitQuad.**PlotRes**(name = '', no_plot = 0, no_plot_err = 0, eps = False, oc_min = True, time_type = 'JD', offset = 2400000, trans = True, title = '', epoch = False, min_type = False, weight = [ ], trans_weight = False, bw = False, double_ax = False, fig_size = None)

Plotting the new values of O-C calculated for values of fitted parameters. It is possible to save figure to png file (eventually also to eps) or only display it on the screen. Outlier points or points with big error could not be plotted. Values of O-C could be displayed in minutes or in days. It is possible to set different type of Julian date displayed on x-label. Date could be transformed by subtraction of given offset. If the date was already transformed before, it is necessary to disable transformation of x values (trans = False) but the value of used offset is suitable to set due to the label on x axis. It is also possible to use epochs on x axis or also have two axis with Julian dates and epochs together. In the case, if the values of O-C of primary and also secondary minima are available, it is possible to distinguished it using the parameter min_type = True. Primary minima are displayed be full circle and the secondary by empty circle. If the weights of points are available, they could be shown on the figures by entering them to weight. Weights are divided into these categories: 0-3, 3-5, 5-8, 8-10. In the case, that the weights are not in this interval, it is necessary to transform them by setting trans_weight = True. If the errors of data points were setted up during initialization of the class and now the weights are given to weight, there will be plotted figure on according to given weights without errorbars.

Parameters:

name: string, optional

Name of file (without extension) where the figure will be saved.

no_plot: integer, optional

Number of outlier points which will not be plotted.

no_plot_err: integer, optional

Number of points with the biggest errors which will not be plotted.

eps: boolean, optional

Save figure also to eps file. (It is necessary to set the name to variable name!)

oc_min: boolean, optional

Values of O-C display in minutes, not in days.

time_type: string, optional

Type of Julian date (JD, HJD, BJD).

offset: float, optional

Value of offset of dates due to shortening values on x axis.

trans: boolean, optional

Transform x values by the offset.

title: string, optional

Title of the figure.

epoch: boolean, optional

Axis x in epochs.

min_type: boolean, optional

Distinguishing primary and secondary minima.

weight: numpy.array, optional

Weights of points.

trans_weight: boolean, optional

Transformation of weights to the range 0 – 10.

bw: boolean, optional

Black and white figure.

double_ax: boolean, optional

Double x axis - dates + epochs.

fig_size: tuple, optional

Custom figure size - e.g. (12,6).

4.2.14 SaveOC

OCFit.FitLinear.**SaveOC**(name, weight = [ ])
OCFit.FitQuad.**SaveOC**(name, weight = [ ])

Saving original values of O-C calculated from given linear ephemeris to file. The data are in columns: observed times of minima, epochs, values of O-C, errors (if given) or weight (if given).

Parameters:

name: string

Name of output file.

weight: numpy.array, optional

Weights of data points.

4.2.15 SaveRes

OCFit.FitLinear.**SaveRes**(name, weight = [ ])
OCFit.FitQuad.**SaveRes**(name, weight = [ ])

Saving residue (new values of O-C) calculated from values of fitted parameters to file. The data are in columns: observed times of minima, epochs, values of O-C, errors (if given) or weight (if given).

Parameters:

name: string

Name of output file.

weight: numpy.array, optional

Weights of data points.

4.3 Example

Example of work with both classes with simulated data. Data from file is simply to load using function numpy.loadtxt.

```python
from OCFit import FitQuad, FitLinear
import numpy as np

#generate data
E = np.arange(0, 100, 1)  #epochs
#simulation of observed times
t = 1e-5*E**2+15*E + 1540.4 + np.random.normal(scale = 0.01, size = E.shape)
err = 0.01*np.ones(E.shape)  #errors of 'observed' times

#usage of FitLinear to first estimation of linear ephemeris
#initialization using estiamtion (original values) of linear ephemeris
#for long time range of observation, the value of period has to be close
#to the right value, otherwise the primary and secondary minims could be mixed
lin = FitLinear(t, 1540, 15.01, err = err)
oc = lin.oc  #O-C calculated from original ephemeris
lin.FitLinear()  #linear fit
lin.FitRobust()  #fitting using robust regression

lin.Summary() #summary of parameters

err = lin.CorrectErr() #re-normalization of errors
oc = lin.FitRobust()  #fitting using robust regression

lin.Summary() #summary of parameters

#plotting figures
#plot of original values of O-C with fit, without transformation of x axis
lin.Plot(trans = False)
#plot of residual O-C, without transformation of x axis
lin.PlotRes(trans = False)

#usage of class FitQuad
#as estimation of linear ephemeris the resault from FitLinear are used
#new values of O-C are only fitted
quad = FitQuad(t, lin.t0, lin.P, oc = oc, err = err)
oc = quad.oc  #O-C calculated from original ephemeris
quad.FitQuad()  #quadratic fit
quad.FitRobust()  #fitting using robust regression

quad.Summary() #summary of parameters

#plotting figures
#plot of original values of O-C with fit, without transformation of x axis
quad.Plot(trans = False)
#plot of residual O-C, without transformation of x axis
quad.PlotRes(trans = False)
```

5 OCFit

The class for fitting O-C according to different models: LiTE for the 3^{rd} and also the 4^{th} body, combination of mass-transfer and LiTE for the 3^{rd} and also the 4^{th} body, apsidal motion and models based on Agol et al. (2005).

5.1 Initialization

OCFit.**OCFit**(t, oc, err = [ ])

For initialization these classes, it is necessary to give observed times of minima (O) and the values of O-C. The values of O-C could be obtained using the classes FitLinear and FitQuad. It is also possible to set the errors of O-C.

Parameters:

t: numpy.array or list

Observed times of minima.

oc: numpy.array or list

Values of O-C in days.

err: numpy.array or list, optional

Errors of O-C.

5.2 Setting the parameters for fitting

OCFit.OCFit.?

It is necessary to set some parameters of model and fitting routine after initialization. They are mainly the limits and steps of parameters, values of fixed parameters, the list of parameters for fitting and model of O-C which will be used for fitting.

Parameters:

limits: dictionary

Limits of fitted parameters. It is necessary to set the list with lower and upper limit for each fitted parameter. The names of parameters are said in description of specific model.

steps: dictionary

Steps of fitted parameters. It is necessary to set the size of step for each fitted parameter. The names of parameters are said in description of specific model.

fit_params: list

The list of parameters which will be fitted. The names of parameters are said in description of specific model.

params: dictionary, optional

The values of fixed parameters. The names of parameters are said in description of specific model.

model: string, optional

Setting model which will be used for fitting of O-C. The name of model is same as name of function for calculation it. The default model is LiTE3.

5.3 Working with weights of data points

If it is possible, it is better to work directly with errors of data points and no with weights! In the case of work with weight (w), the errors could be compute as $1/w$. But in many cases, these errors are really over-estimated.

During working with weights is very good to follow these instruction:

1. during initialization set errors as $1/w$
2. fitting using FitGA
3. re-normalization of errors using AddWeight
4. fitting by MC method using FitMCMC
5. in the saving data by function SaveRes enter the weights in parameter weight
6. in the plotting figures enter weights in parameter weight and eventually use also trans_weight = True

5.4 General notes to the work with the class OCFit

The base of successful fitting is to choose appropriate model. It is useful to minimize the number of model parameter and also number of fitted parameter. It means, not use model with a lot of parameters while many of them are fixed, if it is not really necessary. Using of such complicated model unnecessarily extends computing time and sometimes also decrease a quality of results. Because of this fact, it is sometimes more appropriate to work with residue and not with original data, e. g. for fitting O-C influenced by LiTE from 3^{rd} and also 4^{th} body. Eventually, firstly remove linear / quadratic trend using functions from FitLinear and FitQuad and in the next step fit more complicated changes. It is also useless to fit a parameter which has near zero value, has no influence on shape of model curve and its value is on level of its uncertainty. Good example is quadratic therm Q in the models LiTE3Quad and LiTE34Quad. In the case, that there is not quadratic but only linear trend in data it is useful to fix it on value 0. Its removing from fitted parameters improve precision of the fitting and now it is also possible to decrease a number of iterations of fitting routines.

The second key point is to set suitable interval of parameters for fitting. In many cases, they could be wide enough. However, their reduction increase precision of fitting and reduce a number of necessary iterations. The values of eccentricity e and longitude of pericenter w could be found in whole logically possible intervals, i. e. from 0 to 1 for eccentricity and from 0 to 2π

for longitude of pericenter. For other parameters (such as period of 3^{rd} body P3), it is possible to estimate the interval from the O-C diagram. On the other hand, the value of linear ephemeris t0 and P should be set really small interval because the shape of O-C diagram is very sensitive to value of linear ephemeris. Very wide interval often leads to completely wrong results of fitting. Depending on specific O-C diagram the width of the searching intervals for linear ephemeris should not exceed 1-2 hours. It is the best, if it is possible, avoid fitting of complex models together with linear ephemeris.

For successful fitting, it is even necessary to set sufficient number of iterations for each of fitting algorithms. When genetic algorithms (GA) are used, it should be sufficient to set a number of generations and a size of one generation to $\gtrsim 100-200\times$ *number of fitted parameters.* Like some universal settings, it could be used to set both to the value 1000. For the first test of fitting with GA, it is enough to set both parameters to 100. In a case of Monte Carlo method, the sufficient number of iterations depends on quality of input data, precision of started values and type of model, it is mainly in the range from 10^5 to 10^7. Some universal value should be 10^6. Many times, it is needful to remove at least 10^3 of the first values and use binning with the size $10-100$. For the first estimation, it is enough to use only 1000 iterations without removing values and with unity binning.

5.5 Available functions

5.5.1 AvailableModels

OCFit.OCFit.**AvailableModels**()

Displaying available model of O-C. The list with names of all models is also saved in variable availableModels.

5.5.2 ModelParams

OCFit.OCFit.**ModelParams**(model = '', allModels = False)

Displaying the parameters of selected model of O-C or all available models. If no parameter is given, the parameters of model set in variable model are displayed.

Parameters:

model: string, optional

Name of selected model.

allModels: boolean, optional

Display parameters of all available models.

5.5.3 LiTE

OCFit.OCFit.**LiTE**(t, a_sin_i3, e3, w3, t03, P3)

Function for calculation of O-C based on model of Light-time effect (LiTE). Details are in Irwin (1952). It is not function for fitting model but only auxiliary function!

Parameters:

t: numpy.array

Times for calculation the values of O-C.

a_sin_i3: float

Value of $a \sin i3$ in AU.

e3: float

Eccentricity of the orbit of the 3^{rd} body.

w3: float

Longitude of pericenter of orbit of the 3^{rd} body in radians.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body in days.

Output: numpy.array

Values of O-C calculated from LiTE model.

5.5.4 AgolInPlanet

OCFit.OCFit.**AgolInPlanet**(t, P, a, w, e, mu3, r3, w3, t03, P3)

Function for calculation of O-C based on model of perturbations caused by inner planet according to paper Agol et al. (2005, their eq. (12)). Set model = 'AgolInPlanet' for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time t). Detail description how to set these values is written in a part 5.2.

Parameters:

t: numpy.array

Times for calculation the values of O-C.

P: float

Period of transiting exoplanet in days.

a: float

Semi-mayor axis of orbits of transiting exoplanet in AU.

w: float

Longitude of pericenter of transiting exoplanet in radians.

e: float

Eccentricity of the orbit of transiting exoplanet.

mu3: float

Reduced mass of the 3^{rd} body (inner planet).

r3: float

Radius of the orbit of the 3^{rd} body.

w3: float

Longitude of pericenter of the 3^{rd} body.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body in days.

Output: numpy.array

Values of O-C calculated from model.

5.5.5 AgolInPlanetLin

OCFit.OCFit.**AgolInPlanetLin**(t, t0, P, a, w, e, mu3, r3, w3, t03, P3)

Function for calculation of O-C based on model of perturbations caused by inner planet according to paper Agol et al. (2005, their eq. (12)) combined with the linear model of O-C. The function Epoch has to be used before using this model. Set model = 'AgolInPlanetLin' for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time t). Detail description how to set these values is written in a part 5.2.

Parameters:

t: numpy.array

Times for calculation the values of O-C.

t0: float

Time of reference transit.

P: float

Period of transiting exoplanet in days.

a: float

Semi-mayor axis of orbits of transiting exoplanet in AU.

w: float

Longitude of pericenter of transiting exoplanet in radians.

e: float

Eccentricity of the orbit of transiting exoplanet.

mu3: float

Reduced mass of the 3^{rd} body (inner planet).

r3: float

Radius of the orbit of the 3^{rd} body.

w3: float

Longitude of pericenter of the 3^{rd} body.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body in days in days.

Output: numpy.array

Values of O-C calculated from model.

5.5.6 AgolExPlanet

OCFit.OCFit.**AgolExPlanet**(t, P, mu3, e3, t03, P3)

Function for calculation of O-C based on model of perturbations caused by exterior planet according to paper Agol et al. (2005, their eq. (25)). Set model = 'AgolExPlanet' for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time t). Detail description how to set these values is written in a part 5.2.

Parameters:

t: numpy.array

Times for calculation the values of O-C.

P: float

Period of transiting exoplanet in days.

mu3: float

Reduced mass of the 3^{rd} body (exterior planet).

e3: float

Eccentricity of the orbit of the 3^{rd} body.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body.

Output: numpy.array

Values of O-C calculated from model.

5.5.7 AgolExPlanetLin

OCFit.OCFit.**AgolExPlanetLin**(t, t0, P, mu3, e3, t03, P3)

Function for calculation of O-C based on model of perturbations caused by exterior planet according to paper Agol et al. (2005, their eq. (25)) combined with the linear model of O-C. The function Epoch has to be used before using this model. Set model = 'AgolExPlanetLin' for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time t). Detail description how to set these values is written in a part 5.2.

Parameters:

t: numpy.array

Times for calculation the values of O-C.

t0: float

Time of reference transit.

P: float

Period of transiting exoplanet in days.

mu3: float

Reduced mass of the 3^{rd} body (exterior planet).

e3: float

Eccentricity of the orbit of the 3^{rd} body.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body in days.

Output: numpy.array

Values of O-C calculated from model.

5.5.8 LiTE3

OCFit.OCFit.**LiTE3**(t, a_sin_i3, e3, w3, t03, P3)

Function for calculation of O-C based on model of LiTE caused by presence of the 3^{rd} body. Details are in Irwin (1952). Set model = 'LiTE3' for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time t). Detail description how to set these values is written in a part 5.2.

Parameters:

t: numpy.array

Times for calculation the values of O-C.

a_sin_i3: float

Value of $a \sin i3$ in AU.

e3: float

Eccentricity of the orbit of the 3^{rd} body.

w3: float

Longitude of pericenter of orbit of the 3^{rd} body in radians.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body in days.

Output: numpy.array

Values of O-C calculated from model.

5.5.9 LiTE3Quad

OCFit.OCFit.**LiTE3Quad**(t, t0, P, Q, a_sin_i3, e3, w3, t03, P3)

Function for calculation of O-C based on model of LiTE caused by presence of the 3^{rd} body combined with the quadratic model of O-C. Details are in Irwin (1952). The function Epoch has to be used before using this model. Set model = 'LiTE3Quad' for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time t). Detail description how to set these values is written in a part 5.2.

Parameters:

t: numpy.array

Times for calculation the values of O-C.

t0: float

Time of reference minimum.

P: float

Period of Eclipsing binary in days.

Q: float

Quadratic term.

a_sin_i3: float

Value of $a \sin i3$ in AU.

e3: float

Eccentricity of the orbit of the 3^{rd} body.

w3: float

Longitude of pericenter of orbit of the 3^{rd} body in radians.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body in days.

Output: numpy.array

Values of O-C calculated from model.

5.5.10 LiTE34

OCFit.OCFit.**LiTE34**(t, a_sin_i3, e3, w3, t03, P3, a_sin_i4, e4, w4, t04, P4)

Function for calculation of O-C based on model of LiTE caused by presence of the 3^{rd} and the 4^{th} body. Details are in Irwin (1952). Set `model = 'LiTE34'` for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time `t`). Detail description how to set these values is written in a part 5.2.

t: numpy.array

Times for calculation the values of O-C.

a_sin_i3: float

Value of $a \sin i3$ in AU.

e3: float

Eccentricity of the orbit of the 3^{rd} body.

w3: float

Longitude of pericenter of orbit of the 3^{rd} body in radians.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body in days.

a_sin_i4: float

Value of $a \sin i4$ in AU.

e4: float

Eccentricity of the orbit of the 4^{th} body.

w4: float

Longitude of pericenter of orbit of the 4^{th} body in radians.

t04: float

Time of pericenter passage of the 4^{th} body.

P4: float

Period of the 4^{th} body in days.

Output: numpy.array

Values of O-C calculated from model.

5.5.11 LiTE34Quad

OCFit.OCFit.**LiTE34Quad**(t, t0, P, Q, a_sin_i3, e3, w3, t03, P3, a_sin_i4, e4, w4, t04, P4)

Function for calculation of O-C based on model of LiTE caused by presence of the 3^{rd} and the 4^{th} body combined with the quadratic model of O-C. Details are in Irwin (1952). The function Epoch has to be used before using this model. Set model = 'LiTE34Quad' for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time t). Detail description how to set these values is written in a part 5.2.

Parameters:

t: numpy.array

Times for calculation the values of O-C.

t0: float

Time of reference minimum.

P: float

Period of Eclipsing binary in days.

Q: float

Quadratic term.

a_sin_i3: float

Value of $a \sin i3$ in AU.

e3: float

Eccentricity of the orbit of the 3^{rd} body.

w3: float

Longitude of pericenter of orbit of the 3^{rd} body in radians.

t03: float

Time of pericenter passage of the 3^{rd} body.

P3: float

Period of the 3^{rd} body in days.

a_sin_i4: float

Value of $a \sin i4$ in AU.

e4: float

Eccentricity of the orbit of the 4^{th} body.

w4: float

Longitude of pericenter of orbit of the 4^{th} body in radians.

t04: float

Time of pericenter passage of the 4^{th} body.

P4: float

Period of the 4^{th} body in days.

Output: numpy.array

Values of O-C calculated from model.

5.5.12 Apsidal

OCFit.OCFit.**Apsidal**(t, t0, P, w0, dw, e, min_type)

Function for calculation of O-C based on the model of apsidal motion. Details are in Giménez & Bastero (1995). The function Epoch has to be used before using this model. Set model = 'Apsidal' for using this model. The names of parameters for fitting, setting limits, steps and values of parameters are same same as the name of input parameters of this function (except of time t and types of minima min_type). Detail description how to set these values is written in a part 5.2.

Parameters:

t: numpy.array

Times for calculation the values of O-C.

t0: float

Time of reference minimum.

P: float

Period of Eclipsing binary in days.

w0: float

Initial longitude of pericenter in radians.

dw: float

Angular velocity of line of apsides in radians per period.

e: float

Eccentricity of the orbit.

min_type: numpy.array

Type of minima (0 = primary, 1 = secondary).

Output: numpy.array

Values of O-C calculated from model.

5.5.13 Model

OCFit.OCFit.**Model**(t = [], params = {}, min_type = [])

Function for calculation of selected model of O-C based on given set of parameters in given times. Parameters are given using dictionary and their names are same as names used for setting the model (see descriptions of individual models). If some of parameters of this function is not given, the corresponding parameter from the class will be used.

Parameters:

t: numpy.array or float, optional

Times for calculation the values of O-C.

params: dictionary, optional

Set of model parameters.

min_type: numpy.array or float, optional

Type of minima (0 = primary, 1 = secondary).

Output: numpy.array

Values of O-C calculated from model.

5.5.14 KeplerEQ

OCFit.OCFit.**KeplerEQ**(M, e, eps = 1e-10)

Function for solving the Kepler's equation. Newton-Raphson iteration scheme is used for solving. The expression S_9 from the paper Odell & Gooding (1986) is used as started value. The solution converge for all values of e and M. Computing time starts rapidly raise for $e \gtrsim 0.9$. Because of it, the function KeplerEQMarkley is used in this case.

Parameters:

M: numpy.array or float

Values of mean anomaly in radians.

e: float

Eccentricity of the orbit.

eps: float, optional

Required accuracy of the solution.

Output: numpy.array or float

Values of eccentric anomaly in radians.

5.5.15 KeplerEQMarkley

OCFit.OCFit.**KeplerEQMarkley**(M, e)

Function for solving the Kepler's equation. Non-iterative algorithm from the paper Markley (1995) is used for solving. Computing time is nearly same, not depending on the given values of e and M. The calculation using this algorithm is faster like using function KeplerEQ for $e \gtrsim 0.9$ but it is better to use the function KeplerEQ for small values of eccentricity.

Parameters:

M: numpy.array or float

Values of mean anomaly in radians.

e: float

Eccentricity of the orbit.

Output: numpy.array or float

Values of eccentric anomaly in radians.

5.5.16 Epoch

OCFit.OCFit.**Epoch**(t0, P, t = [ ])

Calculation of epoch of all observed minima. This function has to be used before fitting the model containing linear or quadratic model or model of apsidal motion, eventually plotting the figures with x axis in epochs. Obtained epochs are saved to variable epoch.

Parameters:

t0: float

Time of reference minimum.

P: float

Period in days.

t: numpy.array or float, optional

Times of minima.

Output: numpy.array

Calculated epochs.

5.5.17 PhaseCurve

OCFit.FitLinear.**PhaseCurve**(P, t0, plot = False)

Calculation of phase curve on the basis of given linear ephemeris. The final phase curve could be display on a figure.

Parameters:

P: float

Period.

t0: float

Time of reference minimum.

plot: boolean, optional

Plot also a figure or only calculated phase curve.

Output: numpy.array

Phases and values of O-C.

5.5.18 FitGA

OCFit.OCFit.**FitGA**(generation, size, mut = 0.5, SP = 2, plot_graph = False, visible = True, n_thread = 1, db = ")

Fitting values of O-C using genetic algorithms (GA). For using this function, it is necessary to set limits and steps of all fitted parameters (details are in a part 5.2). Fitting using GA is good to obtain the first estimation of values of parameters which could be used for fitting using MC by function FitMCMC.

It is necessary to set suitable number of generations and their size in variables generation and size. Some hints, how to set them, could be found in a part 5.4. Total number of the conducted calculation of model is generation $\times$ size. Fitting routine could be improved by setting correct probability of mutations and value of selective pressure.

It is possible to save obtained set of individuals in all generations to file and analyse it using function InfoGA.

More detailed informations about using GA could be found in Hartmann & Rieger (2002) or Weise (2011).

Parameters:

generation: integer

Number of generations.

size: integer

Size of one generation, i. e. the number of individual in one generation.

mut: float, optional

Probability of mutations (in the range from 0 to 1).

SP: float, optional

Value of selective pressure (details in Razali & Geraghty (2011)).

plot_graph: boolean, optional

Plotting figure with progress of χ^2 error in individual generations.

visible: boolean, optional

Display a progress of fitting.

n_thread: integer, optional

Number of thread for multithreading.

db: string, optional

Name of the file for saving the fitting.

Output: dictionary

Values of fitted parameters.

5.5.19 InfoGA

OCFit.OCFit.**InfoGA**(db, eps = False)

Function for detail analysis of results of GA obtained using FitGA. For each of fitted parameters, the figure displaying its values through individual generations is generated.

The figure with progress of the χ^2 error in individual generations and the file with summary information about fitting are also created.

Parameters:

db: string

Name of the file with saved fitting.

eps: boolean, optional

Save figures also in eps format.

5.5.20 FitMCMC

OCFit.OCFit.**FitMCMC**(n_iter, burn = 0, binn = 1, visible = True, db = '')

The function for fitting O-C diagram using Markov chain Monte Carlo (MCMC) method. It is necessary to have installed package pymc (Patil et al., 2010). Running of this function is possible only if the initial values of fitted parameters were calculated. They could be obtained from fitting using function FitGA or by manually setting them in params (details in a part 5.2).

It is necessary to set suitable number of iterations in a variable n_iter. Some hints, how to set them, could be found in a part 5.4. Sometimes, it is also necessary to set some other parameters for MCMC. In case that the initial values of parameters are not close to their optimal values to which MC method converge, it is necessary to set using parameter burn a number of MC steps which are removed. Now, the only values in equilibrium are used for the calculation of mean values and errors. The next problem could be strong correlation between following MC steps. It is possible to reduce it by binning generated sequence of MC steps. The size of one block is setted up through parameter binn.

Obtained MC sampling could be saved to file and analysed by function InfoMCMC.

If the errors of data points were not given during initialization of class, it is necessary to calculate it using function CalcErr. Eventually, if the weights are available, it is necessary to transform them to the errors using function AddWeight. In a case, that fitting using FitGA gives good results but fitting using MCMC method worsen them, it could be problem in over-estimated errors of data points. It is possible to normalise them using CorrectErr.

More details about using MCMC method could be found in Brooks et al. (2011).

Parameters:

n_iter: float

Number of MC steps.

burn: float, optional

Number of MC steps which will be reject from calculation.

binn: float, optional

The size of one block for binning.

visible: boolean, optional

Display progress of fitting and summary of results from pymc.

db: string, optional

Name of file for saving MC sampling.

Output: dictionary, dictionary

Values and errors of fitted parameters.

5.5.21 InfoMCMC

OCFit.OCFit.**InfoMCMC**(db, eps = False, geweke = False)

Function for detail analysis of MCMC sampling obtained by FitMCMC. It will be generated a plot with trace of fitting and common plot with trace and histogram for each of fitted parameters. An also, the file with brief informations about sampling, obtained values and errors of the parameter will be created. It is also possible to plot figures for Geweke analysis (details in Geweke (1992)).

Few common figures for all fitted parameters will be generated, too. There are histograms, deviances, correlation plots and 2D histograms displaying the confidence regions. For 2D histograms, the function ConfInt is used, everyone else are generated by the class TraceAnalysis from PyAstronomy.

The last output is table with correlation coefficients between fitted parameters.

Parameters:

db: string

Name of file with saved MC sampling.

eps: boolean, optional

Save figures also in eps format.

geweke: boolean, optional

Geweke analysis.

5.5.22 CalcErr

OCFit.OCFit.**CalcErr**()

Calculation of errors of input data on a basis of values of fitted parameters. This function could not be used before fitting the data. For all point, the same error will be assigned. The size of error will be set that the value of reduced squared error χ_r^2 will be equal 1.

It is necessary to use this function before fitting using MC method by function FitMCMC if the errors of input data were not given during initialization. It is possible to fit data using genetic algorithms by function FitGA. It is assumed that actual values of parameters give model O-C close to the real data.

Output: numpy.array

Calculated errors.

5.5.23 CorrectErr

OCFit.OCFit.**CorrectErr**()

Re-normalization of given errors of input data on a basis of values of fitted parameters. This function could not be used before fitting the data. The size of error will be set that the value of reduced squared error χ_r^2 will be equal 1. For plotting the figure, original values of errors are used.

It is useful to use this function before fitting using MC method by function FitMCMC if the errors of input data are wrong calculated (under- or over-estimated). It is possible to fit data using genetic algorithms by function FitGA. It is assumed that actual values of parameters give model O-C close to the real data.

Output: numpy.array

Calculated errors.

5.5.24 AddWeight

OCFit.OCFit.**AddWeight**(w)

Adding the weights and normalization of errors on a basis of values of fitted parameters. This function could not be used before fitting the data. The size of error will be set that the value of reduced squared error χ_r^2 will be equal 1.

It is necessary to use this function before fitting using MC method by function FitMCMC. It is possible to fit data using genetic algorithms by function FitGA. It is assumed that actual values of parameters give model O-C close to the real data.

Parameters:

w: numpy.array

Weights of points.

Output: numpy.array

Calculated errors.

5.5.25 Amplitude

OCFit.OCFit.**Amplitude**()

Function for calculation of amplitude of O-C based on the used model and setted up parameters. If the function FitMCMC was used, the uncertainty will be also calculated. Obtained value is saved to variable paramsMore and error to variable paramsMore_err.

Output: dictionary

Calculated amplitude and error.

5.5.26 Summary

OCFit.OCFit.**Sumarry**(name = '')

Summary of values and errors of fitted parameters. It is possible to save output to file or display it on the screen.

Parameters:

name: string, optional

Name of output file.

5.5.27 Chi2

OCFit.OCFit.**Chi2**(params)

Function for calculation of χ^2 error for given set of parameters and setted up model.

Parameters:

params: dictionary

Set of parameters of the model.

Output: float

Calculated χ^2 error.

5.5.28 MassFun

OCFit.OCFit.**MassFun**()

Function for calculation of mass function for models based on LiTE. If the function FitMCMC was used, the uncertainty will be also calculated. Obtained value is saved to variable paramsMore and error to variable paramsMore_err.

Output: dictionary

Calculated mass function and error.

5.5.29 AbsoluteParam

OCFit.OCFit.**AbsoluteParam**(M, i = 90, M_err = 0, i_err = 0)

Calculation of mass of the 3^{rd} body and semi-mayor axes for models based on LiTE or Agol's models. If the function FitMCMC was used, the uncertainties will be also calculated. Obtained values are saved to variable paramsMore and errors to variable paramsMore_err.

Parameters:

M: float

Mass of eclipsing binary.

i: float

Inclination of orbit of the 3^{rd} body.

M_err: float

Uncertainty of mass of eclipsing binary.

i_err: float

Uncertainty of orbital inclination of the 3^{rd} body.

Output: dictionary

Values of calculated parameters and errors.

5.5.30 ParamsApsidal

OCFit.OCFit.**ParamsApsidal**()

Calculation of another parameters of the system for the model of apsidal motion. If the function FitMCMC was used, the uncertainties will be also calculated. Obtained values are saved to variable paramsMore and errors to variable paramsMore_err.

Output: dictionary

Values of calculated parameters and errors.

5.5.31 Plot

OCFit.OCFit.**Plot**(name = '', no_plot = 0, no_plot_err = 0, params = {}, eps = False, oc_min = True, time_type = 'JD', offset = 2400000, trans = True, title = '', epoch = False, min_type = False, weight = [ ], trans_weight = False, model2 = False, with_res = False, bw = False, double_ax = False, legend = [ ], fig_size = None)

Plotting original values of O-C and curve of O-C calculated from the fit. It is possible to calculate model curve also for different values of parameters. Parameters are entered to params, it is a dictionary with parameters of the model. Eventually, it is possible to plot curve calculated from fitted values of parameters together with curve calculated from given set of parameters, set model2 = True to do this. In this case, green line is fitted model and red is given model.

It is possible to save figure to png file (eventually also to eps) or only display it on the screen. Outlier points or points with big error could not be plotted. Values of O-C could be displayed in minutes or in days. It is possible to set different type of Julian date displayed on x-label. Date could be transformed by subtraction of given offset. If the date was already transformed before, it is necessary to disable transformation of x values (trans = False) but the value of used offset is suitable to set due to the label on x axis. It is also possible to use epochs on x axis or also have two axis with Julian dates and epochs together. In the case, if the values of O-C of primary and also secondary minima are available, it is possible to distinguished it using the parameter min_type = True. Primary minima are displayed be full circle and the secondary by empty circle. If the weights of points are available, they could be shown on the figures by entering them to weight. Weights are divided into these categories: 0-3, 3-5, 5-8, 8-10. In the case, that the weights are not in this interval, it is necessary to transform them by setting trans_weight = True. If the errors of data points were setted up during initialization of the class and now the weights are given to weight, there will be plotted figure on according to given weights without errorbars.

Parameters:

name: string, optional

Name of file (without extension) where the figure will be saved.

no_plot: integer, optional

Number of outlier points which will not be plotted.

no_plot_err: integer, optional

Number of points with the biggest errors which will not be plotted.

params: dictionary, optional

Parameters of the model.

eps: boolean, optional

Save figure also to eps file. (It is necessary to set the name to variable name!)

oc_min: boolean, optional

Values of O-C display in minutes, not in days.

time_type: string, optional

Type of Julian date (JD, HJD, BJD).

offset: float, optional

Value of offset of dates due to shortening values on x axis.

trans: boolean, optional

Transform x values by the offset.

title: string, optional

Title of the figure.

epoch: boolean, optional

Axis x in epochs.

min_type: boolean, optional

Distinguishing primary and secondary minima.

weight: numpy.array, optional

Weights of points.

trans_weight: boolean, optional

Transformation of weights to the range $0 - 10$.

model2: boolean, optional

Plot model curve of O-C for given and fitted values of parameters.

bw: boolean, optional

Black and white figure.

double_ax: boolean, optional

Double x axis - dates + epochs.

with_res: boolean, optional

Common figure with residue.

legend: list, optional

List of labels for legend (give '', if the label could not be displayed; the 2^{nd} model given trough params is the last one).

fig_size: tuple, optional

Custom figure size - e.g. (12,6).

5.5.32 PlotRes

OCFit.OCFit.**PlotRes**(name = '', no_plot = 0, no_plot_err = 0, params = {}, eps = False, oc_min = True, time_type = 'JD', offset = 2400000, trans = True, title = '', epoch = False, min_type = False, weight = [ ], trans_weight = False, bw = False, double_ax=False, fig_size = None)

Plotting residue (new values of O-C) between original values of O-C and O-Cs calculated from the fit. It is possible to calculate residue also for different set of parameters. Parameters are entered to params, it is a dictionary with parameters of the model.

It is possible to save figure to png file (eventually also to eps) or only display it on the screen. Outlier points or points with big error could not be plotted. Values of O-C could be displayed in minutes or in days. It is possible to set different type of Julian date displayed on x-label. Date could be transformed by subtraction of given offset. If the date was already transformed before, it is necessary to disable transformation of x values (trans = False) but the value of used offset is suitable to set due to the label on x axis. It is also possible to use epochs on x axis or also have two axis with Julian dates and epochs together. In the case, if the values of O-C of primary and also secondary minima are available, it is possible to distinguished it using the parameter min_type = True. Primary minima are displayed be full circle and the secondary by empty circle. If the weights of points are available, they could be shown on the figures by entering them to weight. Weights are divided into these categories: 0-3, 3-5, 5-8, 8-10. In the case, that the weights are not in this interval, it is necessary to transform them by setting trans_weight = True. If the errors of data points were setted up during initialization of the class and now the weights are given to weight, there will be plotted figure on according to given weights without errorbars.

Parameters:

name: string, optional

Name of file (without extension) where the figure will be saved.

no_plot: integer, optional

Number of outlier points which will not be plotted.

no_plot_err: integer, optional

Number of points with the biggest errors which will not be plotted.

params: dictionary, optional

Parameters of the model.

eps: boolean, optional

Save figure also to `eps` file. (It is necessary to set the name to variable `name`!)

oc_min: boolean, optional

Values of O-C display in minutes, not in days.

time_type: string, optional

Type of Julian date (JD, HJD, BJD).

offset: float, optional

Value of offset of dates due to shortening values on x axis.

trans: boolean, optional

Transform x values by the offset.

title: string, optional

Title of the figure.

epoch: boolean, optional

Axis x in epochs.

min_type: boolean, optional

Distinguishing primary and secondary minima.

weight: numpy.array, optional

Weights of points.

trans_weight: boolean, optional

Transformation of weights to the range 0 – 10.

bw: boolean, optional

Black and white figure.

double_ax: boolean, optional

Double x axis - dates + epochs.

fig_size: tuple, optional

Custom figure size - e.g. (12,6).

5.5.33 Save

OCFit.OCFit.**Save**(path)

Saving the parameters of the class OCFit to file. It is possible to use the class OCFitLoad or function Load for loading them, again.

Parameters:

path: string

Name of the file to save parameters of the class.

5.5.34 Load

OCFit.OCFit.**Load**(path)

Loading the parameters of the class OCFit saved by function Save from the file. It is also possible to use the class OCFitLoad with the name of file as only one input parameters. It includes all function of the class OCFit.

Parameters:

path: string

Name of the file with saved parameters of the class.

5.5.35 SaveModel

OCFit.OCFit.**SaveModel**(name, E_min = None, E_max = None, n = 1000, params = {}, t0 = None, P = None)

Saving model curve of O-C calculated on the basis of the fit to file. It is possible to calculate model curve also for different set of parameters. Parameters are entered to params, it is a dictionary with parameters of the model. The data are in columns: observed times of minima, epochs, model values of O-C.

Parameters:

name: string

Name of output file.

E_min: float, optional

Minimal epoch (if not given, minimal epoch of the data).

E_max: float, optional

Maximal epoch (if not given, maximal epoch of the data).

n: integer, optional

Number of points for calculation of model O-C.

params: dictionary, optional

Set of model parameters.

t0: float, optional

Time of reference minimum (necessary, if t0 is not in model).

P: float, optional

Period of system (necessary, if t0 is not in model).

5.5.36 SaveRes

OCFit.OCFit.**SaveRes**(name, params = {}, t0 = None, P = None, weight = [ ])

Saving residue (new values of O-C) between original values of O-C and O-Cs calculated from the fit to file. It is possible to calculate residue also for different set of parameters. Parameters are entered to params, it is a dictionary with parameters of the model. The data are in columns: observed times of minima, epochs, new values of O-C (residue), errors (if given) or weight (if given).

Parameters:

name: string

Name of output file.

params: dictionary, optional

Set of model parameters.

t0: float, optional

Time of reference minimum (necessary, if t0 is not in model).

P: float, optional

Period of system (necessary, if t0 is not in model).

weight: numpy.array, optional

Weights of data points.

5.6 Example

```python
from OCFit import OCFit, FitLinear
import numpy as np

#generating data
E = np.arange(0, 100, 1)  #epochs
#simulation of observed times, LiTE with amplitude 72 min. and period 1125 d
P = 15
t0 = 1540
t = P*E + t0 + 0.05*np.sin(2*np.pi/(5*P)*E)
    + np.random.normal(scale = 0.01, size = E.shape)
err = 0.01*np.ones(E.shape)  #errors of 'observed' times

#usage of FitLinear for calculation of O-C
#initialization using estiamtion (original values) of linear ephemeris
#for long time range of observation, the value of period has to be close
#to the right value, otherwise the primary and secondary minims could be mixed
lin = FitLinear(t, t0, P, err = err)
oc = lin.oc  #O-C calculated from original ephemeris

#initialization of class OCFit and O-C calculated using FitLinear
fit=OCFit(t,oc,err = err)
fit.Epoch(t0,P)  #calculating epochs

#setting the model and parameters for fitting
fit.model='LiTE3'
fit.fit_params=['a_sin_i3', 'e3', 'w3', 't03', 'P3']
fit.limits={'a_sin_i3': [5, 10], 'e3': [0, 1], 'w3': [0, 2*np.pi],
            't03': [1540, 3000], 'P3': [900, 1300]}
fit.steps={'a_sin_i3': 1e-2, 'e3': 1e-2, 'w3': 1e-2, 't03': 10, 'P3': 10}

#fitting using GA without displaying fitting progress
fit.FitGA(100,100,visible=False)
#sumarry of results after GA
fit.Summary()

#fitting using MCMC without displaying fitting progress
fit.FitMCMC(1e3,visible=False)
#sumarry of results after MC
fit.Summary()

#plotting figure
#figure with original O-C with fit without transformation of x axis
#together with residue and 2nd axis in epochs
fit.Plot(trans = False,with_res=True,double_ax=True)
```

6 OCFitLoad

The class for loading parameters of class OCFit saved by its function Save from file. It includes all function of class OCFit.

6.1 Initialization

OCFit.**OCFitLoad**(path)

For initialization of this class, it is necessary to enter only the name of file in which the parameters of the class OCFit are saved.

Parameters:

path: string

Name of file with saved parameters.

References

Agol, E. et al., 2005, MNRAS, 359, 567

Brooks, S. et al. 2011. Handbook of Markov Chain Monte Carlo. CRC Press.

Gajdoš, P., Parimucha, Š., 2018, in preparation

Geweke J., 1992, in Bernardo J. M. et al., eds, Evaluating the accuracy of sampling-based approaches to calculating posterior moments. Clarendon Press, Oxford, p. 169

Giménez, A., Bastero, M., 1995, ApSS, 226, 99

Hartmann, A. T., Rieger, H. 2002. Optimization Algorithms in Physics. Berlin: Wiley-VCH.

Irwin, J. B., 1952, ApJ, 116, 211

Markley, F. L., 1995, CeMDA, 63, 101

Mikulášek, Z., Zejda, M., 2013, Úvod do studia proměnných hvězd, Brno: munipress

Odell, A. W., Gooding, R. H., 1986, CeMec, 38, 307

Patil, A., Huard, D., Fonnesbeck, C. J. 2010. J Stat Softw., 35, 1.

Razali, N. M., Geraghty, J. 2011. Proc. of the World Congress on Eng., II, 1134.

Weise, T. 2011. Global Optimization Algorithm - Theory and Application. 3rd Ed. Germany: online. http://www.it-weise.de/projects/bookNew.pdf.