
Pyedra

Release 0.3.1

Milagros Colazo

Mar 07, 2021

CONTENTS:

1	Motivation	3
2	Repository and Issues	5
3	Indices and tables	25
	Python Module Index	27
	Index	29

Pyedra is a python library that allows you to fit three different models of asteroid phase functions to your observations.

MOTIVATION

Phase curves of asteroids are very important for the scientific community as they can provide information needed for diameter and albedo estimates. Given the large amount of surveys currently being carried out and planned for the next few years, we believe it is necessary to have a tool capable of processing and providing useful information for such an amount of observational data.

Authors

Milagros Colazo (E-mail: milirita.colazovinovo@gmail.com)

REPOSITORY AND ISSUES

<https://github.com/milicolazo/Pyedra>

2.1 Installation

This is the recommended way to install Pyedra.

2.1.1 Installing with pip

Make sure that the Python interpreter can load Pyedra code. The most convenient way to do this is to use virtualenv, virtualenvwrapper, and pip.

After setting up and activating the virtualenv, run the following command:

```
$ pip install Pyedra
...
```

That should be it all.

2.1.2 Installing the development version

If you'd like to be able to update your Pyedra code occasionally with the latest bug fixes and improvements, follow these instructions:

Make sure that you have Git installed and that you can run its commands from a shell. (Enter *git help* at a shell prompt to test this.)

Check out Pyedra main development branch like so:

```
$ git clone https://github.com/milicolazo/Pyedra.git
...
```

This will create a directory *Pyedra* in your current directory.

Then you can proceed to install with the commands

```
$ cd Pyedra
$ pip install -e .
...
```

2.2 Pyedra's Tutorial

This tutorial is intended to serve as a guide for using Pyedra to analyze asteroid phase curve data.

2.2.1 Imports

The first thing we will do is import the necessary libraries. In general you will need the following: - `pyedra` (*pyedra*) is the library that we present in this tutorial. - `pandas` (*pandas*) this library will allow you to import your data as a dataframe.

Note: In this tutorial we assume that you already have experience using these libraries.

```
[1]: import pyedra
import pandas as pd
```

2.2.2 Upload the data

The next thing we have to do is load our data. Pyedra should receive a dataframe with three columns: `id` (MPC number of the asteroid), `alpha` (α , phase angle) and `v` (reduced magnitude in Johnson's V filter). You must respect the order and names of the columns as they have been mentioned. In this step we recommend the use of `pandas`:

```
df = pd.read_csv('somefile.csv')
```

For this tutorial we will use a preloaded data set offered by Pyedra.

```
[2]: df = pyedra.datasets.load_carbognani2019()
```

Here we show you the structure that your data file should have. Note that the file can contain information about many asteroids, which allows to obtain catalogs of the parameters of the phase function for large databases.

```
[3]: df
[3]:
```

	id	alpha	v
0	85	0.89	7.62
1	85	1.18	7.67
2	85	2.07	7.82
3	85	5.11	8.01
4	85	16.24	8.48
5	85	17.49	8.53
6	85	21.24	8.66
7	208	0.64	9.20
8	208	1.01	9.24
9	208	1.74	9.39
10	208	3.75	9.69
11	208	11.85	9.73
12	208	17.28	9.97
13	208	19.18	9.99
14	236	0.86	8.28
15	236	1.27	8.17
16	236	1.95	8.22
17	236	6.39	8.82
18	236	9.77	8.73
19	236	10.69	8.78
20	236	22.14	9.12
21	236	24.72	9.16

(continues on next page)

(continued from previous page)

22	306	5.45	9.18
23	306	5.83	9.17
24	306	6.34	9.21
25	306	8.00	9.28
26	306	16.03	9.61
27	306	21.23	9.60
28	306	22.85	9.69
29	313	0.57	8.94
30	313	0.99	9.06
31	313	4.74	9.21
32	313	8.37	9.41
33	313	10.50	9.54
34	313	20.33	9.86
35	338	1.54	8.74
36	338	3.86	8.93
37	338	10.89	9.49
38	338	18.93	9.85
39	338	19.25	9.77
40	522	2.00	9.23
41	522	2.19	9.28
42	522	4.07	9.36
43	522	4.99	9.41
44	522	7.40	9.55
45	522	13.67	9.85
46	522	16.24	9.88

2.2.3 Fit your data

Pyedra's main objective is to fit a phase function model to our data. Currently the api offers three different models:

- HG_fit (H, G model): $V(\alpha) = H - 2.5\log_{10}[(1 - G)\Phi_1(\alpha) + G\Phi_2(\alpha)]$
- Shev_fit (Shevchenko model): $V(1, \alpha) = V(1, 0) - \frac{a}{1+\alpha} + b \cdot \alpha$
- HG1G2_fit (H, G₁, G₂ model): $V(\alpha) = H - 2.5\log_{10}[G_1\Phi_1(\alpha) + G_2\Phi_2(\alpha) + (1 - G_1 - G_2)\Phi_3(\alpha)]$

We will now explain how to apply each of them. At the end of this tutorial you will notice that they all work in an analogous way and that their implementation is very simple.

HG_fit

Let's assume that we want to fit the biparametric model H, G to our data set. What we will do is invoke Pyedra's HG_fit function:

```
[4]: HG = pyedra.HG_fit(df)
```

We have already created our catalog of H, G parameters for our data set. Let's see what it looks like.

```
[5]: HG
```

```
[5]:
```

	id	H	error_H	G	error_G	R
0	85	7.492423	0.070257	0.043400	0.035114	0.991422
1	208	9.153433	0.217270	0.219822	0.097057	0.899388
2	236	8.059719	0.202373	0.104392	0.094382	0.914150
3	306	8.816185	0.122374	0.306459	0.048506	0.970628

(continues on next page)

(continued from previous page)

```
4  313  8.860208  0.098102  0.170928  0.044624  0.982924
5  338  8.465495  0.087252 -0.121937  0.048183  0.992949
6  522  8.992164  0.063690  0.120200  0.028878  0.991757
```

```
PyedraFitDataFrame - 7 rows x 6 columns
```

R is the coefficient of determination of the fit

All pandas dataframe options are available. For example, you may be interested in knowing the mean H of your sample. To do so:

```
[6]: HG.H.mean()
```

```
[6]: 8.54851801238607
```

Remember that `HG.H` selects the H column.

```
[7]: HG.H
```

```
[7]: 0    7.492423
     1    9.153433
     2    8.059719
     3    8.816185
     4    8.860208
     5    8.465495
     6    8.992164
     Name: H, dtype: float64
```

The `PyedraFitDataFrame` can also be filtered, like a canonical pandas dataframe. Let's assume that we want to save the created catalog, but only for those asteroids whose id is less than 300. All we have to do is:

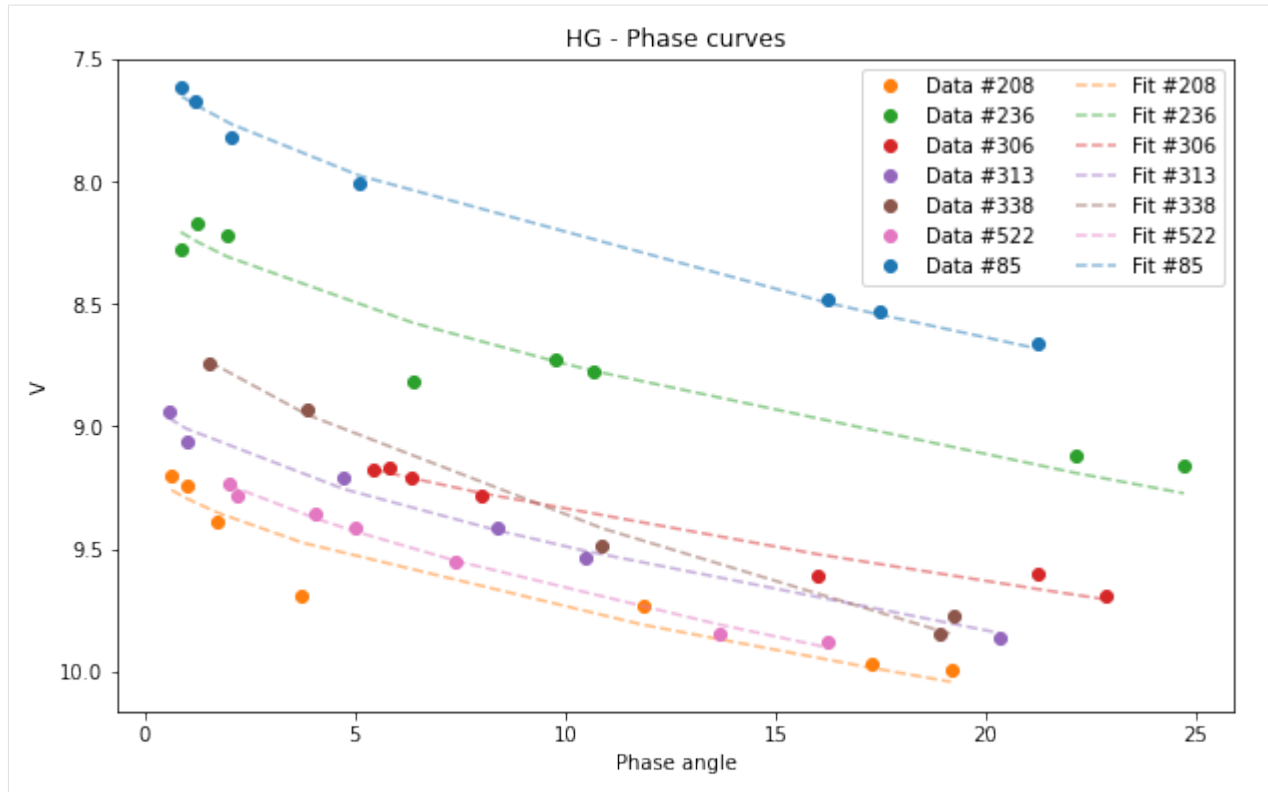
```
[8]: filtered = HG.model_df[HG.model_df['id'] < 300]
     filtered
```

```
[8]:   id      H  error_H      G  error_G      R
     0   85  7.492423  0.070257  0.043400  0.035114  0.991422
     1  208  9.153433  0.217270  0.219822  0.097057  0.899388
     2  236  8.059719  0.202373  0.104392  0.094382  0.914150
```

Finally we want to see our data plotted together with their respective fits. To do this we will use the `.plot` function provided by Pyedra. To obtain the graph with the adjustments of the phase function model we only have to pass to `.plot` the dataframe that contains our data in the following way:

```
[9]: HG.plot(df=df)
```

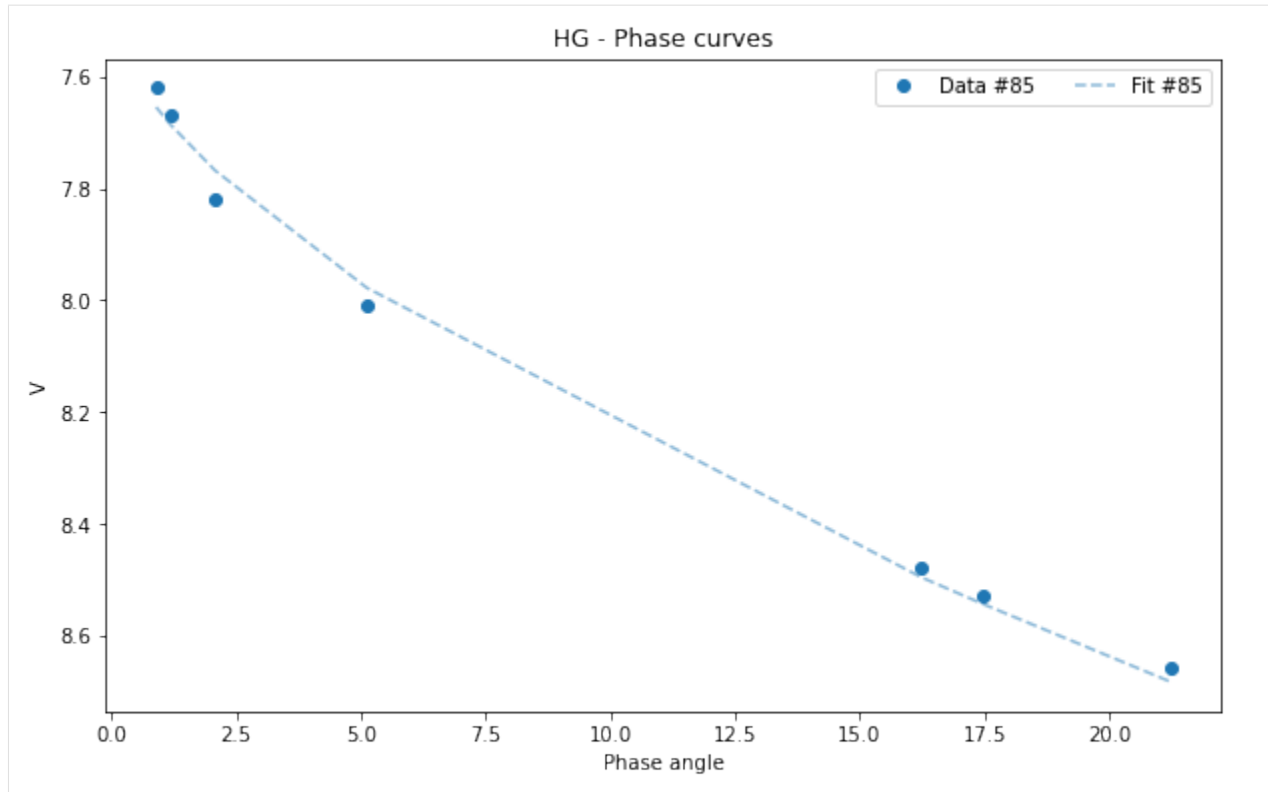
```
[9]: <AxesSubplot:title={'center':'HG - Phase curves'}, xlabel='Phase angle', ylabel='V'>
```



If your database is very large and you want a clearer graph, or if you only want to see the fit of one of the asteroids you can filter your initial dataframe.

```
[10]: asteroid_85 = df[df['id'] == 85]
HG_85 = pyedra.HG_fit(asteroid_85)
HG_85.plot(df = asteroid_85)

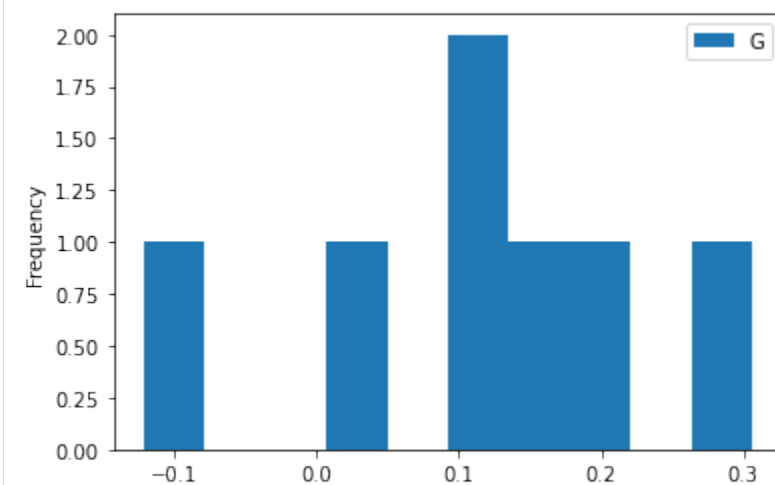
[10]: <AxesSubplot:title={'center':'HG - Phase curves'}, xlabel='Phase angle', ylabel='V'>
```



All pandas plots are available if you want to use any of them. For example, we may want to visualize the histogram of one of the parameters:

```
[11]: HG.plot(y='G', kind='hist')
```

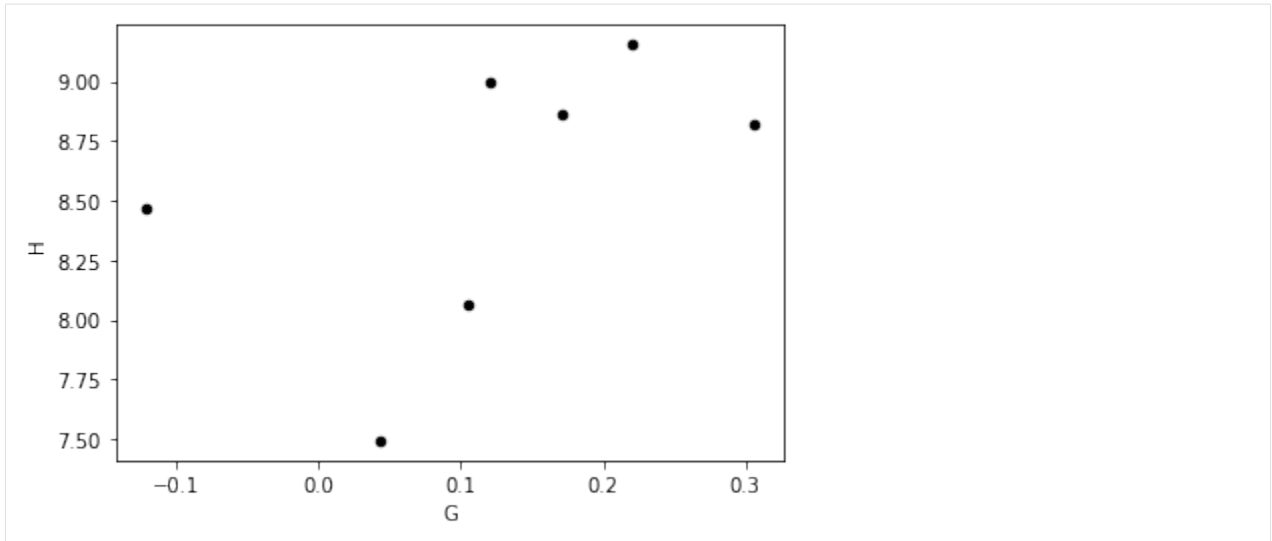
```
[11]: <AxesSubplot:ylabel='Frequency'>
```



Or we may want to find out if there is a correlation between parameters:

```
[12]: HG.plot(x='G', y='H', kind='scatter', marker='o', color='black')
```

```
[12]: <AxesSubplot:xlabel='G', ylabel='H'>
```



Everything we have done in this section can be extended in an analogous way to the rest of the models, as we will see below.

HG1G2_fit

Now we want to adjust the H , G_1 , G_2 model to our data. Use the function `HG1G2_fit` in the following way.

```
[13]: HG1G2 = pyedra.HG1G2_fit(df)
HG1G2
```

```
[13]:
```

	id	H12	error_H12	G1	error_G1	G2	error_G2	R
0	85	7.398776	0.162316	0.303790	0.081963	0.236331	0.062360	0.996285
1	208	8.904819	0.326344	-0.393842	0.320769	0.709746	0.190495	0.976405
2	236	7.901036	0.558052	0.043248	0.361217	0.413350	0.237267	0.934118
3	306	8.224509	1.238547	-0.041959	0.529741	0.367042	0.443392	0.968671
4	313	8.883195	0.347260	0.661550	0.161242	0.127482	0.154199	0.984322
5	338	8.450968	0.391477	0.691141	0.194887	-0.070232	0.173648	0.991939
6	522	9.046202	0.213791	0.705920	0.107933	0.088499	0.087300	0.992124

PyedraFitDataFrame - 7 rows x 8 columns

R is the coefficient of determination of the fit.

We can calculate, for example, the median of each of the columns:

```
[14]: HG1G2.median()
```

```
[14]: id          306.000000
H12           8.450968
error_H12      0.347260
G1             0.303790
error_G1       0.194887
G2            0.236331
error_G2       0.173648
R             0.984322
dtype: float64
```

Again, we can filter our catalog. We are keeping the best settings, that is, those whose R is greater than 0.98.

```
[15]: best_fits = HG1G2.model_df[HG1G2.model_df['R'] > 0.98]
best_fits
```

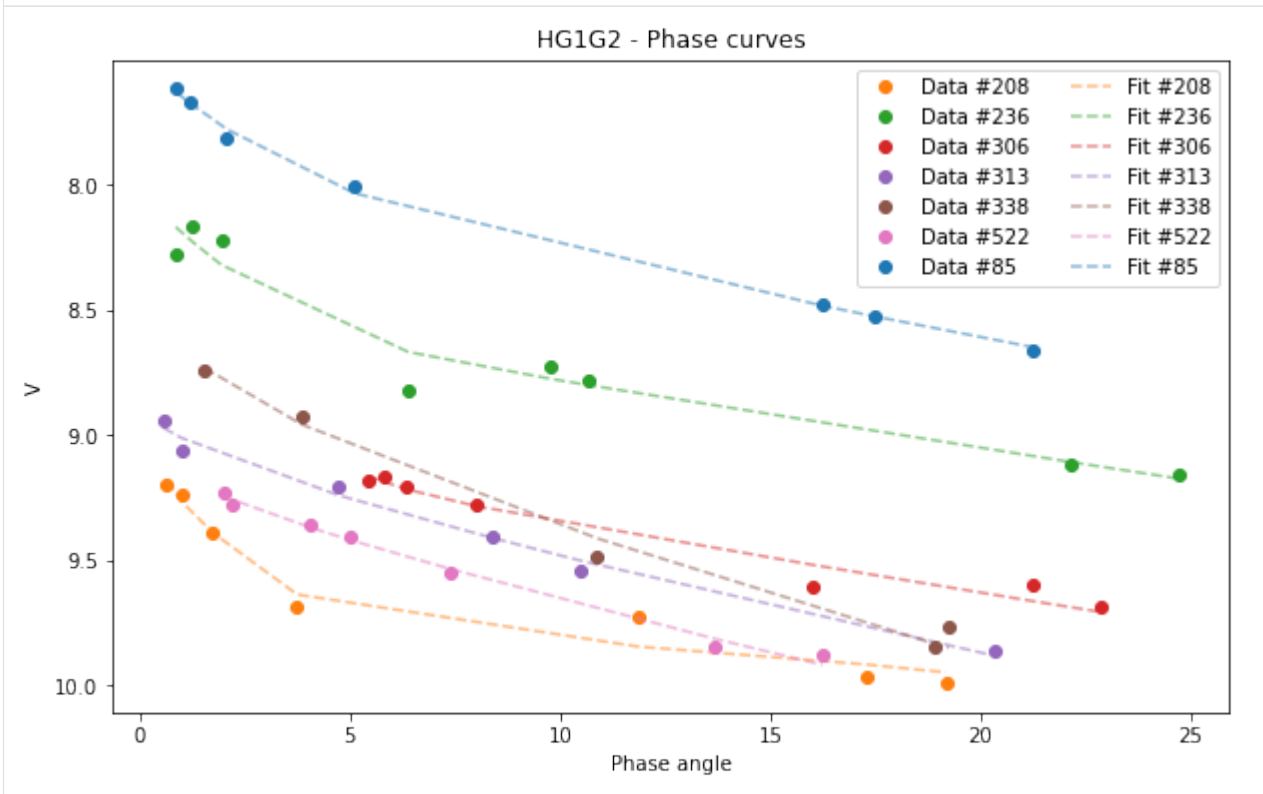
```
[15]:
```

	id	H12	error_H12	G1	error_G1	G2	error_G2	R
0	85	7.398776	0.162316	0.303790	0.081963	0.236331	0.062360	0.996285
4	313	8.883195	0.347260	0.661550	0.161242	0.127482	0.154199	0.984322
5	338	8.450968	0.391477	0.691141	0.194887	-0.070232	0.173648	0.991939
6	522	9.046202	0.213791	0.705920	0.107933	0.088499	0.087300	0.992124

We will now look at the graphics.

```
[16]: HG1G2.plot(df=df)
```

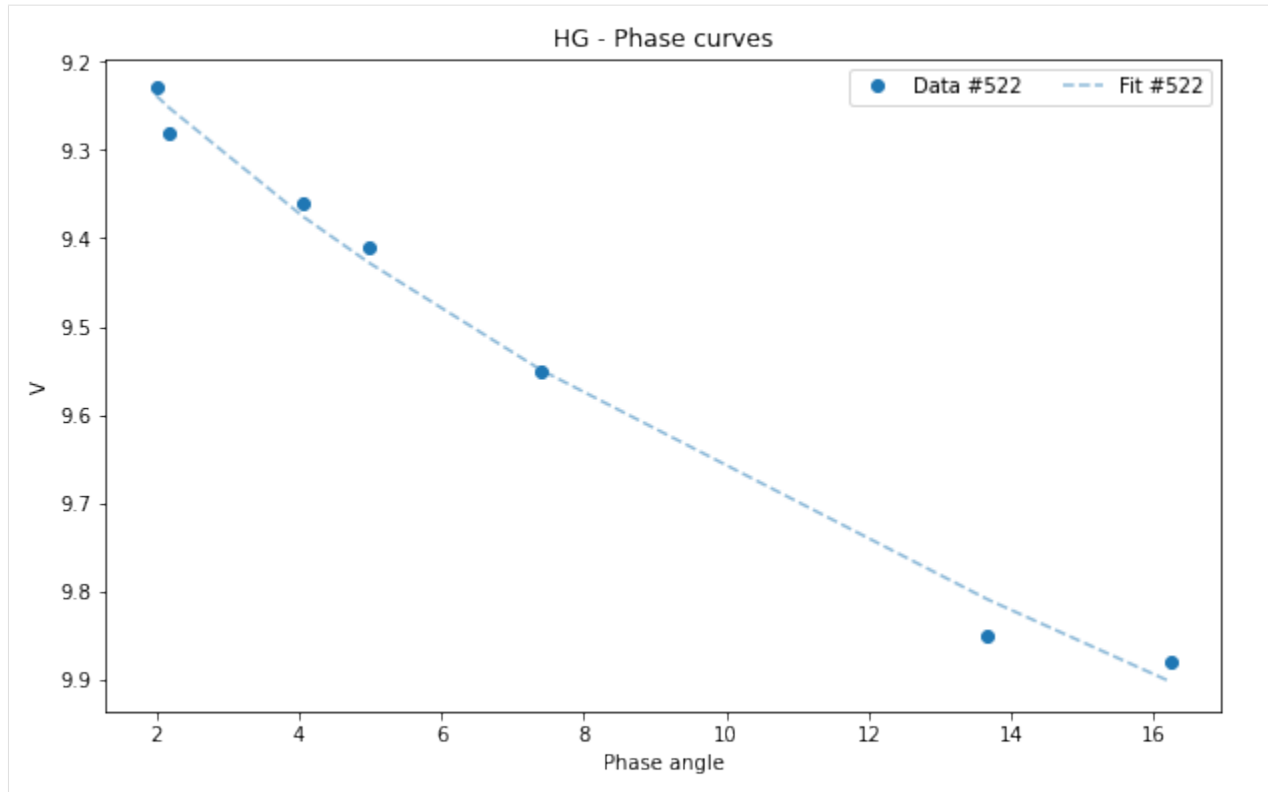
```
[16]: <AxesSubplot:title={'center':'HG1G2 - Phase curves'}, xlabel='Phase angle', ylabel='V'
↪ '>
```



If we want to visualize the graph only of the asteroid (522):

```
[17]: asteroid_522 = df[df['id'] == 522]
HG1G2_522 = pyedra.HG_fit(asteroid_522)
HG1G2_522.plot(df=asteroid_522)
```

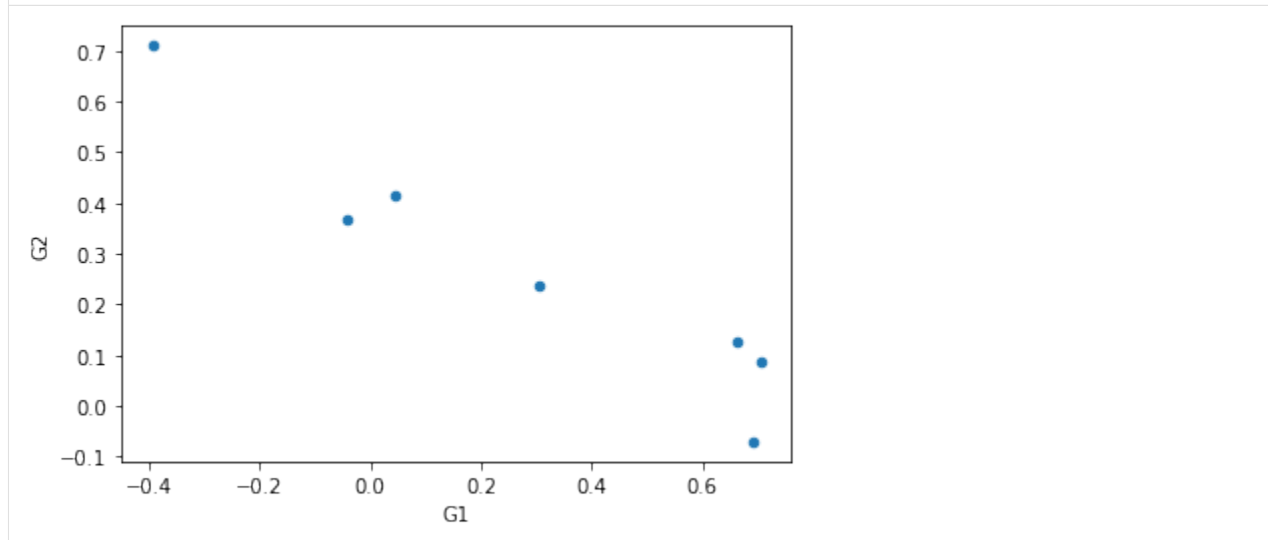
```
[17]: <AxesSubplot:title={'center':'HG - Phase curves'}, xlabel='Phase angle', ylabel='V'>
```



To see the correlation between the parameters G_1 and G_2 we can use the “scatter” graph of pandas:

```
[18]: HG1G2.plot(x='G1', y='G2', kind='scatter')
```

```
[18]: <AxesSubplot:xlabel='G1', ylabel='G2'>
```



Shev_fit

If we want to adjust the Shevchenko model to our data, we must use `Shev_fit`.

```
[19]: Shev = pyedra.Shev_fit(df)
      Shev

[19]:    id    V_lin  error_V_lin      b  error_b      c  error_c  \
0   85  7.957775    0.022576  0.696826  0.051010  0.034637  0.001184
1  208  9.738767    0.146121  0.945126  0.300105  0.014448  0.008524
2  236  8.626281    0.166623  0.916350  0.402758  0.023646  0.008515
3  306  9.600553    0.275835  3.231009  1.549868  0.009097  0.010083
4  313  9.132444    0.071010  0.297612  0.140332  0.037097  0.004712
5  338  9.003280    0.216194  0.900765  0.623240  0.045328  0.011141
6  522  9.292723    0.084461  0.392058  0.267625  0.039625  0.005236

      R
0  0.999542
1  0.960483
2  0.932592
3  0.975662
4  0.988686
5  0.985846
6  0.989935

PyedraFitDataFrame - 7 rows x 8 columns
```

R is the coefficient of determination of the fit.

We can select a particular column and calculate, for example, its minimum:

```
[20]: Shev.V_lin

[20]: 0    7.957775
      1    9.738767
      2    8.626281
      3    9.600553
      4    9.132444
      5    9.003280
      6    9.292723
      Name: V_lin, dtype: float64
```

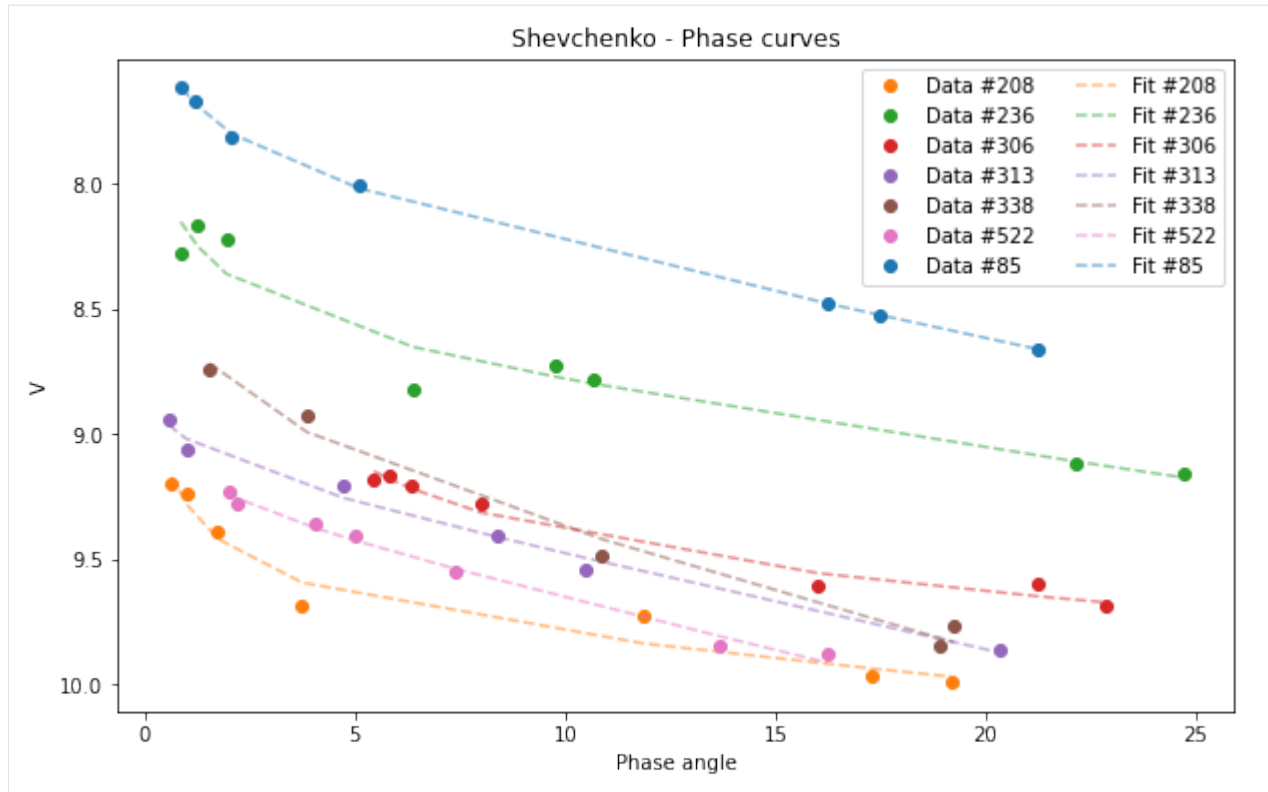
```
[21]: Shev.V_lin.min()

[21]: 7.9577754899895226
```

And obviously we can graph the resulting fit:

```
[22]: Shev.plot(df=df)

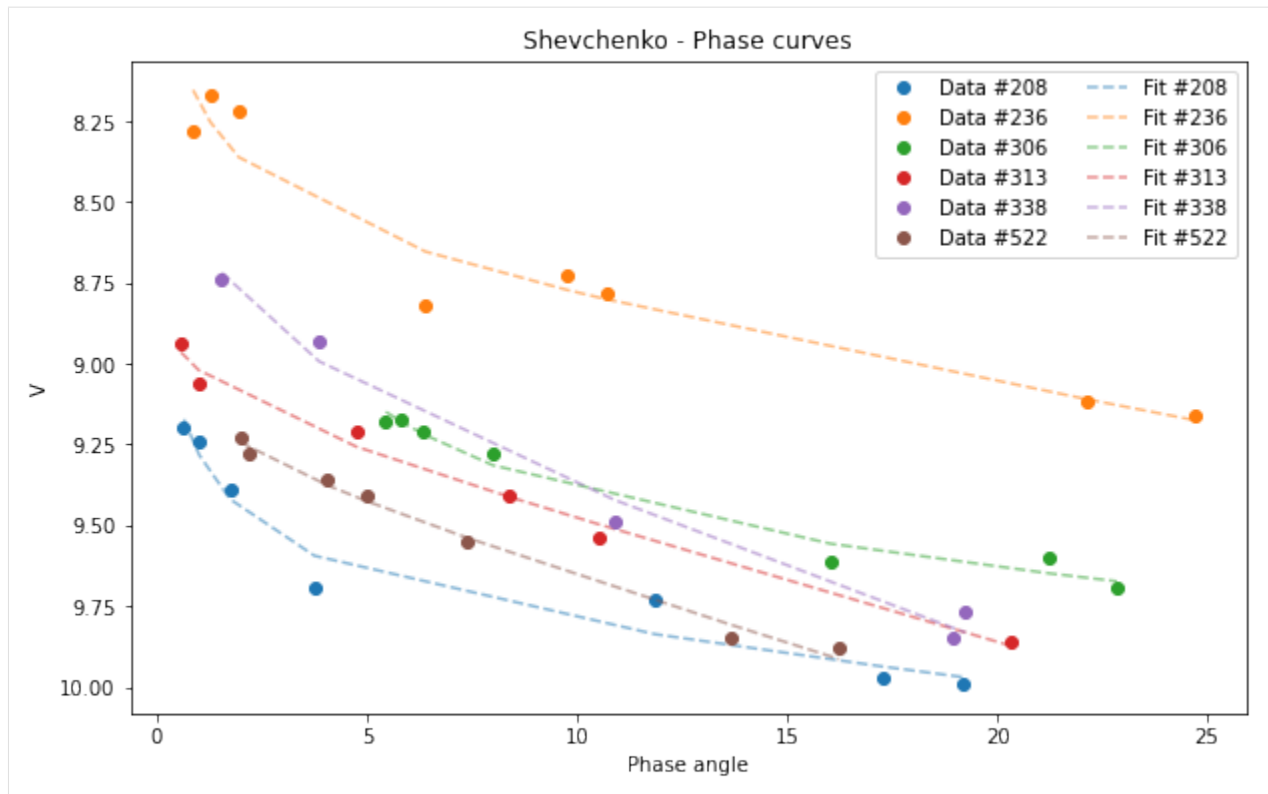
[22]: <AxesSubplot:title={'center':'Shevchenko - Phase curves'}, xlabel='Phase angle',
      ↪ylabel='V'>
```



Selecting a subsample:

```
[23]: subsample = df[df['id'] > 100 ]
      Shev_subsample = pyedra.Shev_fit(subsample)
      Shev_subsample.plot(df=subsample)

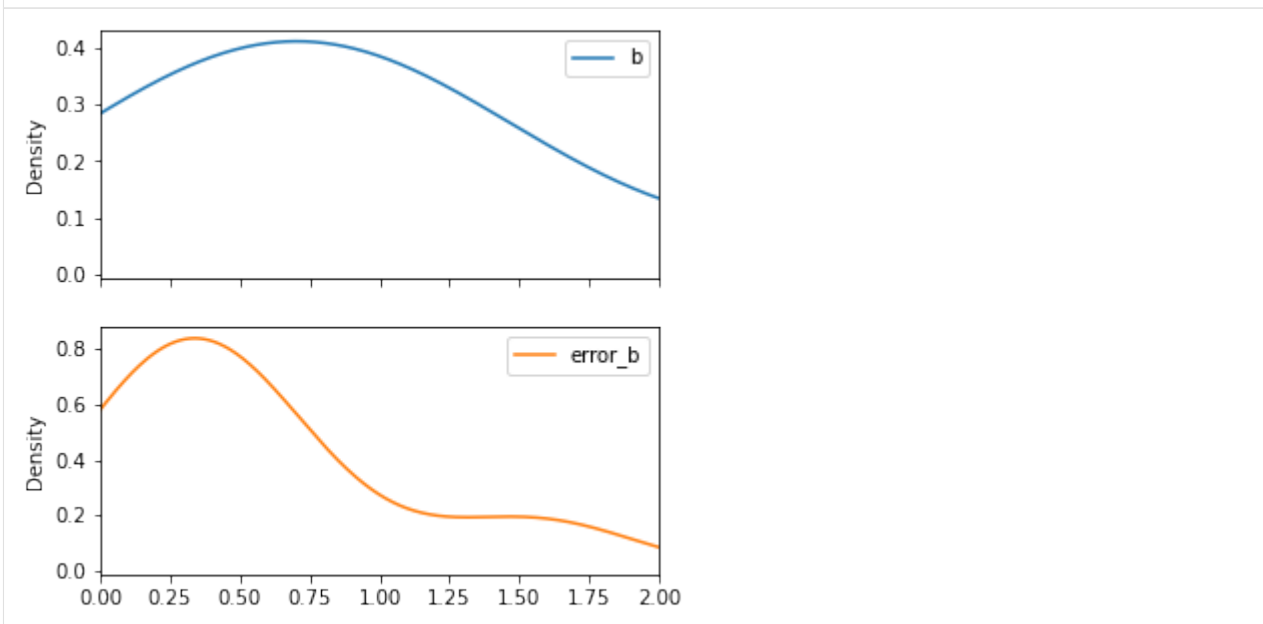
[23]: <AxesSubplot:title={'center':'Shevchenko - Phase curves'}, xlabel='Phase angle',
      ylabel='V'>
```



We can use some of the pandas plot.

```
[24]: Shev_subsample.plot(y=['b', 'error_b'], kind='density', subplots=True, figsize=(5,5),
    ↪xlim=(0,2))
```

```
[24]: array([<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
    dtype=object)
```



2.3 Licence

MIT License

Copyright (c) 2020 Milagros Colazo

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2.4 pyedra API

Pyedra currently has three functions, each of which adjusts a different phase function model.

The input file must be a .csv file with three columns: id (mpc number), alpha (phase angle) and v (reduced magnitude in Johnson’s V filter).

The modules are:

- **HG_fit**: adjusts the H-G biparametric function to the data set.
- **Shev_fit**: adjusts the Shevchenko triparametric function to the data set.
- **HG1G2_fit**: adjusts the triparametric function H-G1-G2 to the data set.

In addition, the data input can be plotted with the chosen fit.

2.4.1 Module `pyedra.core`

Implementation of phase function for asteroids.

```
class pyedra.core.MetaData (data=NOTHING)
    Bases: collections.abc.Mapping
    Implements an immutable dict-like to store the metadata.
    Also provides attribute like access to the keys.
```

Example

```
>>> metadata = MetaData({"a": 1, "b": 2})
>>> metadata.a
1
```

```
>>> metadata["a"]
1
```

class pyedra.core.**PyedraFitDataFrame** (*model, model_df, plot_cls, metadata=NOTHING*)
Bases: object

Initialize a dataframe *model_df* to which we can apply function *a*.

class pyedra.core.**BasePlot** (*pdf*)
Bases: abc.ABC

Plots for HG fit.

abstract property default_plot_kind
Return the default plot to be rendered.

pyedra.core.**obs_counter** (*df, obs, idc='id', alphac='alpha'*)
Count the observations. A minimum is needed to the fits.

Parameters

- **df** (pandas.DataFrame) – The dataframe must with the values
- **idc** (str, optional (default=id)) – Column with the mpc number of the asteroids.
- **alphac** (str, optional (default=alpha)) – Column with the phase angle of the asteroids.
- **obs** (int) – Minimum number of observations needed to perform the fit.

Returns out – Numpy array containing the asteroids whose number of observations is less than *obs*.

Return type ndarray

pyedra.core.**merge_obs** (*obs_a, obs_b, idc_a='id', idc_b='id', alphac_a='alpha', alphac_b='alpha',
magc_a='v', magc_b='v', **kwargs*)
Merge two dataframes with observations.

Sources whose id is not present in *obs_a* are discarded.

The function concatenates two dataframes (*obs_a* and *obs_b*), and assumes that the columns *idc_a*, *alphac_a* and *magc_a* from *obs_a* are equivalent to *idc_b*, *alphac_b* and *magc_b* from a dataframe *obs_b*. The resulting dataframe uses the names of *obs_a* for those three columns and places them at the start, and all other columns of both dataframes combined with the same behavior of `pandas.concat`.

Parameters

- **obs_a** (pandas.DataFrame) – The dataframe must with the observations.
- **obs_b** (pandas.DataFrame) – The dataframe must with the observations to be concatenated to *obs_a*.
- **idc_a** (str, optional (default=id)) – Column with the mpc number of the asteroids of the *obs_a* dataframe.
- **idc_b** (str, optional (default=id)) – Column with the mpc number of the asteroids of the *obs_b* dataframe.

- **alphac_a** (str, optional (default=alpha)) – Column with the phase angle of the asteroids of the obs_a dataframe.
- **alphac_b** (str, optional (default=alpha)) – Column with the phase angle of the asteroids of the obs_b dataframe.
- **magc_a** (str, optional (default=v)) – Column with the magnitude of the obs_a dataframe. The default 'v' value is reference to the reduced magnitude in Johnson's V filter.
- **magc_b** (str, optional (default=v)) – Column with the magnitude of the obs_b dataframe. The default 'v' value is reference to the reduced magnitude in Johnson's V filter.
- **kwargs** (dict or None (optional)) – The parameters to send to the subjacent pandas.concat function.

Returns Merged dataframes.

Return type pd.DataFrame

2.4.2 Module pyedra.hg1g2_model

Implementation of phase function for asteroids.

class pyedra.hg1g2_model.HG1G2Plot (pdf)

Bases: *pyedra.core.BasePlot*

Plots for HG1G2 fit.

curvefit (df, idc='id', alphac='alpha', magc='v', ax=None, cmap=None, fit_kwargs=None, data_kwargs=None)

Plot the phase function using the HG1G2 model.

Parameters

- **df** (pandas.DataFrame) – The dataframe must with the values
- **idc** (str, optional (default=id)) – Column with the mpc number of the asteroids.
- **alphac** (str, optional (default=alpha)) – Column with the phase angle of the asteroids.
- **magc** (str, optional (default=v)) – Column with the magnitude. The default 'v' value is reference to the reduced magnitude in Johnson's V filter.
- **ax** (matplotlib.pyplot.Axis, (optional)) – Matplotlib axis
- **cmap** (None, str or callable (optional)) – Name of the color map to be used (<https://matplotlib.org/users/colormaps.html>). If is None, the default colors of the matplotlib.pyplot.plot function is used, and if, and is a callable is used as colormap generator.
- **fit_kwargs** (dict or None (optional)) – The parameters to send to the fit curve plot. Only label and color can't be provided.
- **data_kwargs** (dict or None (optional)) – The parameters to send to the data plot. Only label and color can't be provided.

Returns The axis where the method draws.

Return type matplotlib.pyplot.Axis

pyedra.hg1g2_model.HG1G2_fit (df, idc='id', alphac='alpha', magc='v')

Fit (H-G1-G2) system to data from table.

HG1G2_fit calculates the H,G1 and G2 parameters of the phase function following the procedure described in⁵

Parameters

- **df** (`pandas.DataFrame`) – The dataframe must with the values
- **idc** (`str`, optional (default=id)) – Column with the mpc number of the asteroids.
- **alphac** (`str`, optional (default=alpha)) – Column with the phase angle of the asteroids.
- **magc** (`str`, optional (default=v)) – Column with the magnitude. The default ‘v’ value is reference to the reduced magnitude in Johnson’s V filter.

Returns The output contains eight columns: id (mpc number of the asteroid), H (absolute magnitude returned by the fit), H error (fit H parameter error), G1 (G1 parameter returned by the fit), G1 error (fit G1 parameter error), G2 (G2 parameter returned by the fit), G2 error (fit G2 parameter error), and R (fit determination coefficient).

Return type `PyedraFitDataFrame`

References

2.4.3 Module `pyedra.hg_model`

H,G model for Pyedra.

class `pyedra.hg_model.HGPlot` (*pdf*)

Bases: `pyedra.core.BasePlot`

Plots for HG fit.

curvefit (*df*, *idc*=‘id’, *alphac*=‘alpha’, *magc*=‘v’, *ax*=None, *cmap*=None, *fit_kwargs*=None, *data_kwargs*=None)

Plot the phase function using the HG model.

Parameters

- **df** (`pandas.DataFrame`) – The dataframe must with the values
- **idc** (`str`, optional (default=id)) – Column with the mpc number of the asteroids.
- **alphac** (`str`, optional (default=alpha)) – Column with the phase angle of the asteroids.
- **magc** (`str`, optional (default=v)) – Column with the magnitude. The default ‘v’ value is reference to the reduced magnitude in Johnson’s V filter.
- **ax** (`matplotlib.pyplot.Axis`, optional) – Matplotlib axis
- **cmap** (None, `str` or callable (optional)) – Name of the color map to be used (<https://matplotlib.org/users/colormaps.html>). If is None, the default colors of the `matplotlib.pyplot.plot` function is used, and if, and is a callable is used as colormap generator.
- **fit_kwargs** (`dict` or None (optional)) – The parameters to send to the fit curve plot. Only `label` and `color` can’t be provided.
- **data_kwargs** (`dict` or None (optional)) – The parameters to send to the data plot. Only `label` and `color` can’t be provided.

Returns The axis where the method draws.

Return type `matplotlib.pyplot.Axis`

⁵ Muinonen K., Belskaya I. N., Cellino A., Delbò M., Lvasseur-Regourd A.-C., Penttilä A., Tedesco E. F., 2010, Icarus, 209, 542.

`pyedra.hg_model.HG_fit(df, idc='id', alphac='alpha', magc='v')`

Fit (H-G) system to data from table.

HG_fit calculates the H and G parameters of the phase function following the procedure described in¹.

Parameters

- **df** (`pandas.DataFrame`) – The dataframe must with the values
- **idc** (`str`, optional (default=`id`)) – Column with the mpc number of the asteroids.
- **alphac** (`str`, optional (default=`alpha`)) – Column with the phase angle of the asteroids.
- **magc** (`str`, optional (default=`v`)) – Column with the magnitude. The default ‘v’ value is reference to the reduced magnitude in Johnson’s V filter.

Returns The output contains 7 columns: id (mpc number of the asteroid), H (absolute magnitude returned by the fit), H error (fit H parameter error), G (slope parameter returned by the fit), G error (fit G parameter error), R (fit determination coefficient) and observations (number of observation of the given asteroid).

Return type `PyedraFitDataFrame`

References

2.4.4 Module `pyedra.shevchenko_model`

Shevchenko model for Pyedra.

class `pyedra.shevchenko_model.ShevPlot` (*pdf*)

Bases: `pyedra.core.BasePlot`

Plots for Shevchenko fit.

curvefit (`df`, `idc='id'`, `alphac='alpha'`, `magc='v'`, `ax=None`, `cmap=None`, `fit_kwargs=None`, `data_kwargs=None`)

Plot the phase function using the Shev model.

Parameters

- **df** (`pandas.DataFrame`) – The dataframe must with the values
- **idc** (`str`, optional (default=`id`)) – Column with the mpc number of the asteroids.
- **alphac** (`str`, optional (default=`alpha`)) – Column with the phase angle of the asteroids.
- **magc** (`str`, optional (default=`v`)) – Column with the magnitude. The default ‘v’ value is reference to the reduced magnitude in Johnson’s V filter.
- **ax** (`matplotlib.pyplot.Axis`, (optional)) – Matplotlib axis
- **cmap** (`None`, `str` or callable (optional)) – Name of the color map to be used (<https://matplotlib.org/users/colormaps.html>). If is `None`, the default colors of the `matplotlib.pyplot.plot` function is used, and if, and is a callable is used as colormap generator.
- **fit_kwargs** (`dict` or `None` (optional)) – The parameters to send to the fit curve plot. Only `label` and `color` can’t be provided.
- **data_kwargs** (`dict` or `None` (optional)) – The parameters to send to the data plot. Only `label` and `color` can’t be provided.

Returns The axis where the method draws.

¹ Muinonen K., Belskaya I. N., Cellino A., Delbò M., Lvasseur-Regourd A.-C., Penttilä A., Tedesco E. F., 2010, Icarus, 209, 542.

Return type `matplotlib.pyplot.Axis`

`pyedra.shevchenko_model.Shev_fit(df, idc='id', alphac='alpha', magc='v')`
Fit Shevchenko equation to data from table.

`Shev_fit` calculates parameters of the three-parameter empirical function proposed by Schevchenko^{2,3}.

Parameters

- **df** (`pandas.DataFrame`) – The dataframe must with the values
- **idc** (`str`, optional (default=`id`)) – Column with the mpc number of the asteroids.
- **alphac** (`str`, optional (default=`alpha`)) – Column with the phase angle of the asteroids.
- **magc** (`str`, optional (default=`v`)) – Column with the magnitude. The default 'v' value is reference to the reduced magnitude in Johnson's V filter.

Returns The output contains six columns: `id` (mpc number of the asteroid), `V_lin` (magnitude calculated by linear extrapolation to zero), `V_lin error` (fit `V_lin` parameter error), `b` (fit parameter characterizing the opposition effect amplitude), `b error` (fit `b` parameter error), `c` (fit parameter describing the linear part of the magnitude phase dependence), `c error` (fit `c` parameter error)⁴ and `R` (fit determination coefficient).

Return type `PyedraFitDataFrame`

References

2.4.5 Module `pyedra.datasets`

The `pyedra.datasets` module includes utilities to load datasets.

It also features some artificial data generators.

`pyedra.datasets.load_carbognani2019()`
Input for use with the phase functions.

This dataset contains the first and second columns of Table 2 of⁶. These columns correspond to: phase angle (°) and `V_max` (mag). `V_max` is the reduced magnitude of the lightcurve maximum.

References

`pyedra.datasets.load_penttila2016()`
Tabulated values of the base functions for H-G1-G2 system.

This dataset corresponds to Table B.4 of⁷.

² Shevchenko, V. G. 1996. Analysis of the asteroid phase dependences of brightness. *Lunar Planet Sci.* XXVII, 1086.

³ Shevchenko, V. G. 1997. Analysis of asteroid brightness phase relations. *Solar System Res.* 31, 219-224.

⁴ Belskaya, I. N., Shevchenko, V. G., 2000. Opposition effect of asteroids. *Icarus* 147, 94-105.

⁶ Carbognani, A., Cellino, A., & Caminiti, S. (2019). New phase-magnitude curves for some main belt asteroids, fit of different photometric systems and calibration of the albedo-Photometry relation. *Planetary and Space Science*, 169, 15-34.

⁷ A. Penttilä, V. G. Shevchenko, O. Wilkman, & K. Muinonen (2016). H,G1, G2 photometric phase function extended to low-accuracy data. 123:117–125.

References

`pyedra.datasets.load_gaia()`
Gaia observations.

The data used to obtain these quantities were downloaded from the Gaia Archive (<https://gea.esac.esa.int/archive/>)⁸.

References

⁸ Gaia Collaboration et al., 2018, A&A, 616, A13

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

- `pyedra.core`, [17](#)
- `pyedra.datasets`, [22](#)
- `pyedra.hg1g2_model`, [19](#)
- `pyedra.hg_model`, [20](#)
- `pyedra.shevchenko_model`, [21](#)

INDEX

B

`BasePlot` (class in `pyedra.core`), 18

C

`curvefit()` (`pyedra.hg1g2_model.HG1G2Plot` method), 19

`curvefit()` (`pyedra.hg_model.HGPlot` method), 20

`curvefit()` (`pyedra.shevchenko_model.ShevPlot` method), 21

D

`default_plot_kind()` (`pyedra.core.BasePlot` property), 18

H

`HG1G2_fit()` (in module `pyedra.hg1g2_model`), 19

`HG1G2Plot` (class in `pyedra.hg1g2_model`), 19

`HG_fit()` (in module `pyedra.hg_model`), 20

`HGPlot` (class in `pyedra.hg_model`), 20

L

`load_carbognani2019()` (in module `pyedra.datasets`), 22

`load_gaia()` (in module `pyedra.datasets`), 23

`load_penttila2016()` (in module `pyedra.datasets`), 22

M

`merge_obs()` (in module `pyedra.core`), 18

`MetaData` (class in `pyedra.core`), 17

module

`pyedra.core`, 17

`pyedra.datasets`, 22

`pyedra.hg1g2_model`, 19

`pyedra.hg_model`, 20

`pyedra.shevchenko_model`, 21

O

`obs_counter()` (in module `pyedra.core`), 18

P

`pyedra.core`

module, 17

`pyedra.datasets`

module, 22

`pyedra.hg1g2_model`

module, 19

`pyedra.hg_model`

module, 20

`pyedra.shevchenko_model`

module, 21

`PyedraFitDataFrame` (class in `pyedra.core`), 18

S

`Shev_fit()` (in module `pyedra.shevchenko_model`), 22

`ShevPlot` (class in `pyedra.shevchenko_model`), 21