Sbpy Application Report

Qingyu Zhang

June 2021

Contents

Chapter 1

Introduction

First document. This is a simple example, with no extra parameters or packages included.

1.1 Sbpy

sbpy is an Astropy affiliated package for small-body planetary astronomy. It is meant to supplement functionality provided by Astropy with functions and methods that are frequently used in the context of planetary astronomy with a clear focus on asteroids and comets [**?**].

sbpy is motivated by the idea to provide a basis of well-tested and well-documented methods to planetary astronomers in order to boost productivity and reproduceability [**?**].

1.2 Module Structure

sbpy consists of a number of sub-modules, each of which provides functionality that is tailored to individual aspects asteroid and comet research. The general module design is shown in the following figure 1.1.

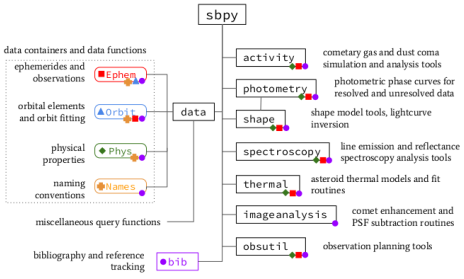

Figure 1.1: the module structure of sbpy

Modules are shown as rectangular boxes, important classes as rounded colored boxes. The left-hand side of the schematic is mainly populated with support modules that act as data containers and query functions. The right-hand side of the schematic shows modules that focus on small body-related functionality. Colored symbols match the colors and symbols of classes and modules they are using.

In our application, we will be using *Ephem* and *Orbit* modules as our data source, then *activity* module to calculate some tasks.

Chapter 2

Computational Task

2.1 Task1 Orbit

2.1.1 Work Flow

The first computational task is to compute the orbit of a planet and its position at designated time.

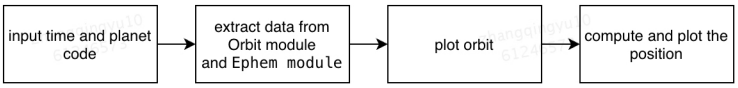

Figure 2.1: task2 workflow

2.1.2 User Input

The application needs user to input the information of the comet or planet including target-id, target-type and so on. Users can get these information through `https://ssd.jpl.nasa.gov/horizons.cgi#top`

2.1.3 Data Extraction

Take the default input as an example, we want to plot the orbit of the *46P*, *Earth* and *Jupiter*. Part of the Orbit data and filed names are shown blow.

```
print(comet.table)
print(planets.table)
 targetname   M1          e          k1 ...         Tp        H   G
                mag                    ...                        
---------- ------ ---------- ---------- ... ------------- ----- ----
46P/Wirtanen 16.5 0.658118229146137 6 4.5 ... 2454499.083390636   0 0.15
46P/Wirtanen 16.5 0.658730684124734 5 4.5 ...  2458465.42938508   0 0.15
  targetname            e                  q          ...  H   G
                                          AU          ...           
---------- ------------- ------------------ ---------- ... ----- ----
  Earth (399) 0.01747538550963633 0.9832960477296679 ...   0 0.15
Jupiter (599)  0.0488828634756756  4.948219304330323 ...   0 0.15

print(comet.field_names)
<TableColumns names=('targetname','M1','e','k1','q','incl','Omega','w','n','M','nu','a','Q','P','epoch','Tp')>
```

Figure 2.2: Orbit Data

2.1.4 Computation and Visualization

Code below demonstrates the core of the orbit plotting

```
1 epochs = orb['epoch'] + np.linspace(-1, 1, 1000) * orb['P
      '] / 2
2 eph = Ephem.from_oo(orb, epochs=epochs, dynmodel='2')
3 ax.plot(eph['x'].value, eph['y'].value, **plot_kwargs)
```

After running the program we will get the orbit plot below. From the outside to the inside, the orbits of Jupiter, comet, and Earth respectively.

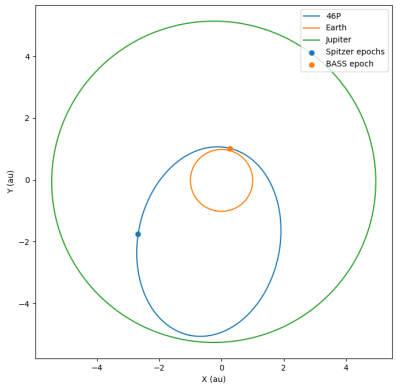

Figure 2.3: Orbit Plot

2.2 Task2 Haser Model

The Haser (1957) model calculates the spatial distribution of coma molecules under photolysis

2.2.1 Work Flow

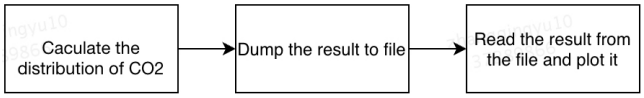

Figure 2.4: Haser model work flow

2.2.2 Computation

Calculate the Haser model column density from 1 to 10^4 km for CO_2, produced at a rate of 10^{28}/s. Let the expansion velocity be 0.8 km/s. Let the heliocentric distance range from $1au$ to $10^{11}au$. Code blow shows the computation of the model.

```
gamma = v * tau * (rh / u.au)**2
co2 = gas.Haser(Q, v, gamma)
```

Then save the result object using pickle.

```
with open('haser_ds', 'wb') as f:
    pk.dump(res, f)
```

2.2.3 Visualization

First We read the result

```
with open('haser_ds', 'rb') as f:
    res = pk.load(f)
rho = res['rho']
data = res['data']
```

Figure blow is the plot of the distribution of CO_2

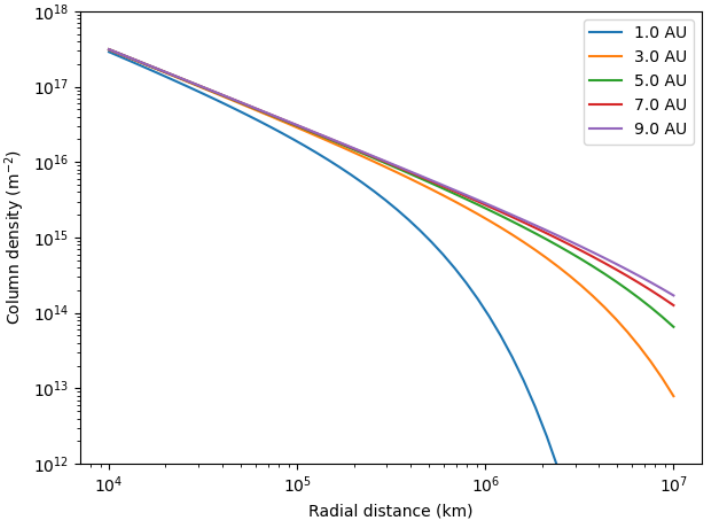

Figure 2.5: Distribution of CO_2

Chapter 3

Conclusion

Sbpy is an useful package to assist researchers to handle observation data and manipulate them. It also supply many modules other than data, which is quite difficult for me to understand their application. I used the data module and the activity module, it broadened my field of vision, I've never had a chance to touch professional Astronomy data, fresh but difficult.

Most Importantly, it helps me getting more familiar with those techniques and tools learned from this course.

Bibliography

[1] Michael Mommert, Michael S. p. Kelley, Miguel de Val-Borro, Jian-Yang Li, Giannina Guzman, Brigitta Sipőcz, Josef Ďurech, Mikael Granvik, Will Grundy, Nick Moskovitz, Antti Penttilä, and Nalin Samarasinha. sbpy: A python module for small-body planetary astronomy. *Journal of Open Source Software*, 4(38):1426, 2019.