Dust Emission

Edgar Ortiz
edgar.ortiz@ua.cl

Abstract

In this work, I present a python library built using OPP to handle the creation of emission spectra from the models described in Draine & Li (2007), using data stored here. Along with the library, I present I working implementation of it in a python script.

Introduction

The interstellar medium is not empty at all, it contains atomic gas (H and He mainly), some molecules and dust. The dust is heated by the visible and ultraviolet part of the EM spectrum, to then reemit the absorbed energy in the infrared. By analyzing and fitting a measure infrared spectrum, we can infer the physical properties of the dust, like grain size and dust chemical components among others.

Drain & Li (2007) proposed that the fitting of the part of the spectra, for a galaxy or a large region due to dust emission, to be the linear sum of the emission generated by dust when heated with a single radiation field from a star, plus the emission generated by the dust when heated by a distribution of starlight radiation, as indicated in the next equation:

j_nu = (1-gamma)*j_nu[umin,umin] + gamma*j_nu[umin,umax] (1)

Here, j_nu[umin,umin] and j_nu[umin,umax] correspond to the emission due to a single radiation field, and a distribution of them respectively. Gamma is a parameter between zero and one.

Drain & Li (2007) provided data "calculated for dust composed of mixtures of amorphous silicate and graphitic grains, including varying amounts of polycyclic aromatic hydrocarbon (PAH) particles." Here they provide the emission for the two starlight radiation distributions appearing in the last equation. In this work, I provide a python code that uses OOP, the last equation and the data provided by Drain & Li (2007) to compute the emission spectrum for different combinations of umim, umax, %PAH, and gamma. I provide a working example of the code, showing a plot of the spectrum generated with a particular combination of these parameters, as well as other derived quantities.

Data

The data Drain & Li (2007) provide, consist of a set of directories, where a folder is named with the format Uumin. Each of these folders contains a set of .txt files with the format: Uumin_umax_model.txt. Here umin and umax refer to the lower and upper bound for the energy in the radiation fields that heat the dust. The model section refers to the dust models, for which there are eleven possibilities as shown in the next table:

Model	**q_PAH**
MW3.1_00	0.47 %
MW3.1_10	1.12 %
MW3.1_20	1.77 %
MW3.1_30	2.50 %
MW3.1_40	3.19 %
MW3.1_50	3.90 %
MW3.1_60	4.58 %
LMC2_00	0.75 %
LMC2_05	1.49 %

LMC2_10	2.37 %
smc	0.10 %

The second column refers to the fraction mass of dust that is made up of PAH particles, containing less than 1000 C atoms.

A .txt file contains the emission spectrum j_nu[umin,umax] between 1cm and 1 micron in units of [Jy cm2 sr-1 H-1]. These emission spectrums will be used with equation (1) to compute a brand new emission spectrum using the parameter gamma, that measures the fraction of dust mass that is heated by the second radiation term in equation (1)

Algorithms

I provide a library named **models_dust** and a script named demonstration.py. The library contains the necessary classes and functions needed to answer the questions in this graded practical. Let's examine each piece of code in models_dust:

1. **Data**: this class, when instantiated, loads all the data from Drain & Li (2007) in memory. Only the data relevant to equation (1) is loaded. The constructor for this class requires only the path to the directory where the data is. This class contains the next methods:
    a. **txt_files()**: uses the **walk** method from the python module **os** to generate the paths to all directories and files relevant to this project. It returns lists. The first list contains the path to all Uumin directories. The second list contains the names of these directories and finally, the third list contains the names of all the files within all the directories.
    b. **arr_dat()**: this method returns a NumPy array with the relevant data of one of the .txt files. If the file is empty, it returns a Numpy array of three zeros.

    c. **dic_files()**: this method returns a dictionary where the key is the directory path and the values are lists with the files in the corresponding directory.
    d. **dic_arr()**: this method returns a dictionary containing all the data, where a key is the name of a .txt file and the value is the array containing the relevant data from this file.
2. **Filter_handler():** this class manages the data in the provided filters and it is initialized with the name of the filter. It contains the next methods:
    a. **lamb_f():** returns a 1D array of wavelengths in nm
    b. **energy():** it returns T (1) in units of energy, such that the integral of T is normalized to one.
    c. **interpolate(interval):** it receives an interval over which the interpolation of the filter data is evaluated. This evaluation is returned by this method.
3. **lamb_inter(arr_1,arr_2)**: returns the sorted union of the 1D arrays it gets as input. Additionally, the elements that are repeated, are striped, letting only one.
4. **Model:** class to create an emission spectrum for a given combination of umin, umax, model & gamma. It is initialized with the four parameters previously discussed. Additionally, it has: the variable **data and filter** which are an instantiated object from the class Data and Filter_handler respectively. It also by default loads the map between a model and its % of mas of PAH particles in the dust. It contains the next methods:
    a. **raw_model():** it returns four variables, the first two correspond to the name and array of data of the .txt file, corresponding to the first term of equation (1). The last two are the same as before but for the second term of equation (1).
    b. **spectrum():** it computes the emission spectrum based on (1). It returns two arrays, the first one is the wavelength range and the second one the emission.

c. **interpolate(interval):** it receives an interval over which the interpolation of the emission data is evaluated. This evaluation is returned by this method.
d. **plot_spec():** it plots the computed spectrum from (1) in a semilog plot (for x), it shows it and saves it in a PDF file as well.
e. **bolometric():** it returns the bolometric luminosity for the emission spectrum computed using (1).
f. **L_density():** it returns the luminosity density in two formats, per wavelength and per frequency.

A working demonstration for the library: demostration.py

In this script, I import everything from the class and then I load all the relevant data in memory using the class Data:

```
D = Data()
```

Then I check it is working for a couple of files:

```
dic_arr = d.dic_arr()
for x,y in dic_arr.items():
    print(x)
    print(y)
    break
print('U0.20_1e3_MW3.1_60.txt')
print(dic_arr['U0.20_1e3_MW3.1_60.txt'])
```

Next, I load the filter data

```
band = 'PACS_red.dat'
filter = Filter_handler(band)
```

Next, I generate one model for the parameters:

- **umin=** 0.20
- **umax=** 1e3

- **model=** MW3.1_60
- **gamma=** 0.3

```
m = Model(umin= '0.20', umax='1e3', model='MW3.1_60',gamma = 0.3,data = d, filter=filter)
```

Now we can obtain the wavelength range and emission for this model

```
l,s = m.spectrum()
print("This is the wavelength range in nm:\n")
print(l)
print("This is the emission in [W/nm/(Kg of H)]\n")
print(s)
```

Now a plot of the spectrum (check for a PDF file in this directory)

```
m.plot_spec()
```

Now we can compute the bolometric luminosity in [W/(Kg of H)]

```
bolometric = m.bolometric()
print(bolometric)
```

Finally, we compute the luminosity density.

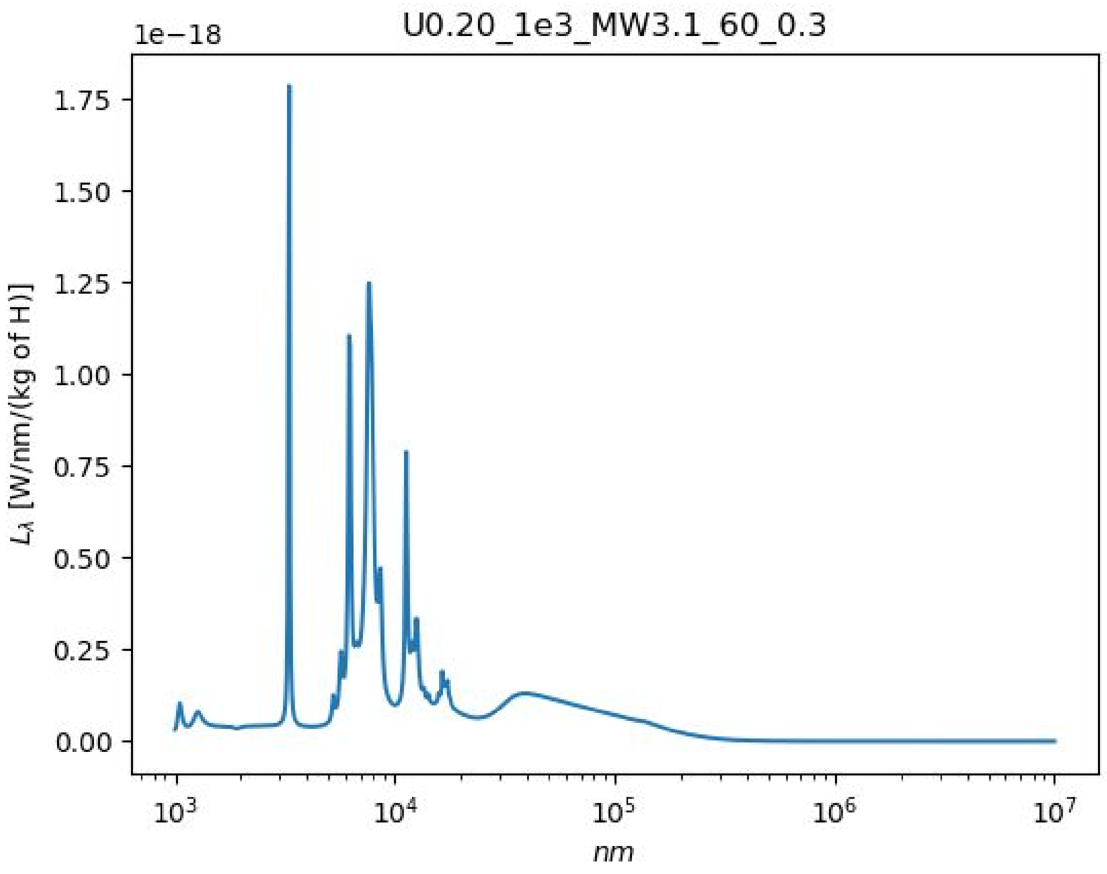
```
w,f = m.L_density()
print("Luminosity density per unit wavelength in [W/nm/(Kg of H)]")
print(w)
print('\n')
print("Luminosity density per unit frequency in [W/Hz/(Kg of H)]")
print(f)
```

That's it!!

Discussion & Results

Several files in the data are empty files. To avoid any break of the code because of that, I generated a Numpy array with 3 zeros to then use with some logical statements to alert the user about this fact. One example corresponds to the file: U0.20_1e2_LMC2_00.txt. The reason for the three zeros is that the relevant data contains three columns (see the readme file in the data link).

For the combination of parameters described at the end of the last section, I obtained the next spectrum:



And some quantities computed were:

- The bolometric luminosity in [W/(Kg of H)]: 1.7929945974158052e-14
- For the filter PACS_red.dat:
    - Luminosity density per unit wavelength in [W/nm/(Kg of H)]: 3.797012714098192e-20
    - Luminosity density per unit frequency in [W/Hz/(Kg of H)]: 1.9347544221733025e-24

The original units in the data were not the ones presented here, so many units conversion had to be implemented.

Conclusions

With demostratio.py, I showed that the classes and functions in the library models_dust do the required work for this assignment. Nonetheless, the test was done on a couple of randomly selected files. There could be the possibility of failure if the format is corrupted in other files.