FU Ori Modeling Code Documentation

Tony Rodríguez

Last updated: 23 August 2019

1 Introduction

This document should be kept as up-to-date as is reasonable. It serves as a main guide to the code functionality and features, but should not be used as a perfectly detailed handbook to know the input/output of every single function. Look directly into the code and accompanying comments for that.

This document should ideally be read with the source code beside to follow along.

2 Jupyter Notebooks

The two main ones are: Jupyter Notebooks/FU Ori General Modeling Interface.ipynb and Jupyter Notebooks/Visualization and Diagnostics/SED General MCMC Fitting.ipynb

2.1 First level, within the Jupyter Notebooks folder

- `FU Ori Accretion Disks.pynb` was the first attempt at using pure blackbodies to compare to KHH88. This uses an old disk model class structure, but works. It's only meant to give a comparison between KHH88 and a first-attempt of this summer's model.
- `FU Ori General Modeling Interface.ipynb` **This is the main notebook for making model spectra and comparing to data**. This makes a model spectrum from stellar atmospheres and is, at the time of writing this, set up to handle Gaia 17bpi. Detailed steps are contained within the notebook in the Markdown format.
- `FU Ori Sub Interface ....ipynb` these three notebooks feature the exact same functionality as the General one above, but are specialized to read in data for Gaia 17bpi or HBC 722 or V1057 Cyg. They're essentially all the same notebook, but the data-reading is very slightly different.

2.2 Second level, within the Jupyter Notebooks/Visualization and Diagnostics folder

- `Annuli Array Creation.ipynb` is just for diagnosing the annuli-creation algorithm.
- `Annuli Binning Visualization.ipynb` is for diagnosing the annuli-creation algorithm visually and has cool plots showing how $T(r)$ is sampled and what the resulting SEDs look like.
- `Data Reading Module.ipynb` is just for diagnosing the data-reading functions and for handling the high-dispersion data that we didn't use over the summer as well.
- `Extinction Tutorial.ipynb` is for diagnosing the extinction function adapted from Cardelli, Claython, Mathis (1989), and has some cool plots showing what A_λ/A_V looks like and how SEDs are modified when extinction is adopted.
- `Gaia 17bpi/HBC 722/V1057 Cyg SED Preparation.ipynb` are used for resampling data points from the spectra to be used as data points of the SED when feeding into the MCMC for parameter estimation. They have relevant plots demonstrating that a polynomial is fit to the spectra and only about a dozen samples are taken from that to be used as part of the SED.
- `Post MCMC Fit.ipynb` is used to tweak parameters after the MCMC fit if it doesn't converge as desired.
- `SED General MCMC Fitting.ipynb` **is the main notebook handling the MCMc fit to determine** $M_*, R_*, \dot{M}$ given different values of i. i is considered a fixed parameter in this notebook, but just make different copies of it and set different values of i to get $M_*, R_*, \dot{M}$ for those different values of i. Thorough documentation is located within this notebook. **Use this one to create blackbody SEDs on the fly.**
- `Spectral Atlas Generator.ipynb` does exactly that: generates spectral atlases of stellar atmospheres at different temperatures across wavelength space.
- `Stellar Atmosphere Testing.ipynb` just ensures I'm reading in stellar atmospheres correctly.
- `Visual of Disk Annuli` is a toy plot for visualization purposes, but Mathematica does this better.

3 External Python Files

3.1 Helper functions: `fu_ori_functions_new.py`

- Constants: These are hard-coded in as global variables. `RAD_MAX_DISK` and `ATMOS_FACTOR` are the only ones that may need to be changed if $T(r)$ and stellar atmosphere files are changed, respectively.
- Temperature functions: These describe $T(r)$ for an accretion disk. They assume that the temperature inside of $R_{\max} = 1.361R_*$, where $T(R_{\max}) = T_{\max}$, is $T_{\max}$. This is to keep this as a physical model and not have the temperature go to zero at the center. Note that the value of 1.361 is hard-coded as a global variable, and if $T(r)$ is changed then this should as well. The accompanying functions are used for minimization of this

and taking in array-like input.

- Annuli-generating functions: These are really the heart of the computation. If something should be changed about the creation of annuli, it should be done here.

  `generateMasterList` is the fundamental function here. It takes the max/min temperatures of a defined disk and calculates where it should create annuli. Given the range of stellar atmospheres specified (as they would exist in the corresponding library), it will create annuli spaced by even temperature outwards from the central star. Once these stellar atmospheres run out (e.g. at 2000 K), then annuli created by even spacing (say, R_*) are created.
- Luminosity functions: The blackbody function returns $\pi \cdot B_\lambda(\lambda, T)$, since the stellar atmospheres are also given with that additional factor of π. This is the factor that comes from Lambert's cosine law that should be included for ideal isotropic radiators. The stellar atmosphere reading function currently only works for the models by Allard in directory format I have. In principle, modifying this function alone should increase flexibility to other models.
- Model spectrum functions: This function takes care of the interpolation of stellar atmospheres.
- Line broadening kernels: These functions store normalized kernels for different types of broadening. Note that the velocity function is separate to allow flexibility for departure from non-Keplerian rotation. Also note that the instrumental broadening kernel is just a toy model at the moment. This should be explore more carefully in the future.
- Extinction functions: Taken directly from Cardelli, Clayton, Mathis (1989), **with a modification** to extend the near-infrared extinciton function out a bit farther into the infrared. This makes the SED look smooth in that regime as it converges to match the non-extincted SED.

3.2 Classes: `fu_ori_classes_new.py`

- The `Annulus` class can be set have a blackbody or stellar atmosphere profile attributed to it. The attributes also include interpolated and broadened luminosities, so it can store both. There is currently a function that broadens the profile by an instrumental kernel, but the kernel in the other .py file is currently a toy model. Note that the stellar atmosphere assigned to an annulus is currently dependent on surface gravity. For the calculated surface gravity, if it is greater than 2.0, a 4.0 stellar atmosphere is assigned to the annulus, and a 1.5 surface gravity stellar atmosphere otherwise.
- The `FUOri` class is the heart of the code.
  1. Given the instance attributes $R_*, R_{\rm inner} = R_*, R_{\rm outer}, M_*, \dot{M}, i,$ and A_V, a class is created.
  2. Based on the range of stellar atmospheres, annuli are created according to a hybrid algorithm, sampling by even temeprature step until atmospheres run out, and by even distance step out to $R_{\rm outer}$.
  3. The annuli are "prepared" in `prepareAnnuli`, which means that their luminosity profiles are interpolated and broadened.

4. A model spectrum is created by multiplying the luminosity density by area of each respective annulus and summing.
5. This model spectrum is now extincted according to A_V and the conversion in Cardelli, Claython, Mathis (1989) to A_λ/A_V.

You can also get the total luminosity of the disk, which is done by taking each annulus as a blackbody and summing up. This should also be done for stellar atmospheres by integrating below L_λ. The fractional flux/luminosity calculation is also handled in this class, **where the outer annuli are weighted** by $\Delta T_{\text{inner, fixed}}/\Delta T_{\text{outer, variable}}$ to create a smooth transition when shifting from even temperature step to even distance step.

- The `Star` class is essentially a modified `Annulus` class, except with a modified emitting area and broadening profile. It also features extinction, which isn't done in the `Annulus` class and saved for the model spectrum creation in `FUOri` when all contributions are already there.