MOSFiT notes

Brenna Mockler

January 2022

1 Quick start

https://mosfit.readthedocs.io/en/latest/getting_started.html provides instructions for starting a first run. I'd recommend trying if first with data already in `MOSFiT` format (e.g. from https://sne.space/) just to check that it's working.

You will probably also want to try running in 'generative mode' (check the docs pages), which does not require transient data. When not given a transient data file, it will randomly sample from the prior ranges in parameters.json.

Running `MOSFiT` from the command line will create a set of new folders named: jupyter, models, modules, and products (the products folder will only be created if you save outputs). The jupyter folder has sample plotting/analysis code in it, the products folder has the saved outputs. One output file is called 'transientname.json' and one is called 'transientname_extras.json' (only saved if you save extra parameters), and then two more that are exact copies of those files called 'walkers.json' and 'extras.json' respectively.

Another plug for https://mosfit.readthedocs.io – it covers a lot of what I cover here, and goes into more depth on subjects such as changing models, so definitely check it out!

2 Converting data to `MOSFiT` format

Pass `MOSFiT` your data file as you would a transient name or transient data file already in `MOSFiT` format. This will trigger `MOSFiT` to start formatting it for you, asking you questions along the way. It will eventually produce a new data file in `MOSFiT` format titled 'transientname.json'. `MOSFiT` will automatically start fitting the transient afterwards, but I recommend cancelling that fit to add more info to the new data file first.

Data to add to new file:

- milky way reddening correction E(B - V), parameter name 'ebv'. If the data is already corrected for MW reddening, then set this to zero. To determine the right formatting for this and any other parameters you add to the new data file, I recommend checking out other mosfit data files, e.g. ones on https://sne.space/. It's pretty straightforward, as new parameters

are added to the file as python dictionary keys and values, but I think it's helpful to look at other files for reference.

- redshift, parameter name 'redshift'.
- luminosity distance, parameter name 'lumdist'. This should be automatically calculated using the redshift, but I've had issues with it not getting calculated so I think it's worth it to just add it to the file. You can use astropy's cosmology module to calculate this from redshfit (from astropy.cosmology import Planck15 as cosmo; cosmo.luminosity_distance(redshift)).

3 Changing the basic models

The easiest way to make changes to the models is using the 'model.json' (actually called tde.json, kilonova.json, etc) and 'parameters.json' files. You can find the parameters.json file in 'models/modelname/' where 'modelname' is just the `MOSFiT` model used (e.g. tde, kilonova, etc). The model.json files are in the `MOSFiT` package, which has a similar directory structure with a 'models/modelname/' directory tree. You can either copy it over from wherever you stored the `MOSFiT` package, or just download a new one from github: https://github.com/guillochon/MOSFiT/tree/master/mosfit/models. The files in the model folders in your run directory will override the files in the model folders in the `MOSFiT` package directories, so this is an easy way to change models without changing the code in the package.

The parameters.json file lists all the parameters that are currently being varied over. You can easily change the minimum and maximum values for parameters here. If you don't want a parameter to be varied over, you can remove it from this file. Then the parameter will take on the default value in the relevant model.json file (or the defaults.json file if it's a default parameter in all models, such as nhhost).

The model.json file first lists all the parameters used in the model that aren't default parameters, and then lists the modules that make up the model. The modules are the code pieces that build up the model such as an 'engine' component that calculates luminosity, or an 'sed' component that converts from bolometric luminosity to luminosity at different wavelengths. When listing the parameters, it also sets default values. It should be noted that these values are not necessarily average values for their respective transient population, so they shouldn't be taken as default model values for fits/example light curves without double checking that they make sense first. You should be able to change them to whatever you like when using them as default model values (e.g. when removing variable parameters from parameters.json).

I think you can similarly change the parameter values in the default.json file, but I don't remember doing this so I'm not sure if it entails anything else. Alternatively, you can just set any given parameter to a constant for a single run using the '–fix-parameters' or '-F' command line argument (as discussed below

for nhhost). In general, if you want to have `MOSFiT` do something differently, I'd check out the command line arguments page first as there is a lot of functionality there: https://mosfit.readthedocs.io/en/latest/command_line.html.

4 Other things to keep in mind

- If you're fitting data, I think it's often better to have `MOSFiT` fit for host extinction itself while fitting for everything else (with the parameter 'nhhost'). However, you can set it to whatever you want using the the command line argument '-F' (e.g. '-F nhhost 1e20'). If the data is already corrected for host extinction, you should force this parameter to zero ('-F nhhost 0'). You could alternatively make nhhost a constant parameter, but this is a bit more complicated (especially since nhhost is a default parameter, not one specific to individual transient models).
- The output data files have all the info from the original data file, and then they also have info for the output 'walkers'. This is true when running mcmc fits, and also when just running in 'generative' mode and randomly sampling parameter values from the prior ranges.