CARB Data Notes

Kenny Van August 23, 2021

Introduction

This file is meant to serve as a README/introduction to the python scripts I have written to analyze the data used in the Convection and Rotation Boosted (CARB) magnetic braking paper. You can find the code at the following github link: HERE

Directory Structure

The directory structure of the data was setup to be split into clear subdirectories so it is easier to search the simulations. The directories are first split in period based on the density of the period spacing. Following this, the directories are split into groups of 10 to 12 donor masses to allow for multiple groups of donor masses to be analyzed using the python code at once. Each of the individual donor mass directories contain the appropriate initial period directories. The structure is shown below in the following tree:

```
Main Directory
├── Fine Period (−0.6 ≤ log10 P ≤ 1.64)
│   └── Grouped Mass Directories
│       └── Individual Donor Masses
│           └── Individual Binary Periods
└── Coarse Period (1.65 ≤ log10 P ≤ 4)
    └── Grouped Mass Directories
        └── Individual Donor Masses
            └── Individual Binary Periods
```

The majority of the code is written to be run in the `Main Directory` and the code itself is written assuming the above directory structure.

Calculating the "Duty Cycle"

In our work we calculate a value which we refer to internally in the python code as the duty cycle in `calc_duty_cycle.py`. This value is given by

$$\text{Duty Cycle} = \frac{\text{Observed Time}}{\text{Detectable Time}} \tag{1}$$

These two times refer to the following:

- Observed time : The amount of time a model appears similar to an observed LMXB
- Detectable time : The amount of time the model is deemed to be detectable. A system is considered detectable if it is classified as persistent using a disk instability model or if it is within the persistent mass transfer rate.

This code is run within a given grouped mass directory and requires the name, orbital period, mass ratio, mass transfer rate, and effective temperature value of an observed LMXB to compare to simulations. An additional file labelled as a `dc_template.dat` is also parsed which contains all of the initial orbital periods and donor masses. Due to the code being run in the grouped mass directory, the user can effectively parallelize the calculation by running multiple instances of `calc_duty_cycle.py`. Once an instance of `calc_duty_cycle.py` is run in the grouped mass directories, they can be combined by running `combine_dc.py` in the main directory. This process will result in a file containing the duty cycles of each observed LMXB of interest.

dc_template.dat

The `dc_template.dat` file is generated by `gen_dc_file.py`. This code will generate the appropriate mass - period combinations based on user inputs. In principle this mass - period generation could be folded into the `calc_duty_cycle.py` code but this additional step was purposely made to ensure that the proper combinations were generated.

Progenitor Figures

To generate these figures we use a python code that handles the duty cycle files generated from `calc_duty_cycle.py` and `combine_dc.py`. This code is named `NDensity.py` and is imported into the following simpler files

- `plot_NDensity_class.py`: This script generates a figure in $\log_{10} P - M_d$ showing the progenitors of observed LMXBs colour coded based on user set groupings. In our case we split the LMXBs into UCXB, short, medium and long period systems.
- `plot_many_NDensity.py`: This script generates a figure in $\log_{10} P - M_d$ showing the progenitors of observed LMXBs split into 4 subplots based on the matching observed LMXB. The colour of plot shows the fraction of time the simulation appears similar to an observed LMXB given by the following equation.

$$f_{\text{obs},i} = \frac{\text{Time matching a specific observed system}}{\text{Time satisfying persistent condition}} \tag{2}$$

- `plot_short_NDensity.py`: This script generates a figure in $\log_{10} P - M_d$ showing the progenitors of the short LMXBs split into the 4 individual systems. The colour of plot shows the fraction of time the simulation appears similar to an observed LMXB.
- `plot_2_NDensity.py`: This script generates a figure in $\log_{10} P - M_d$ showing the progenitors of the medium LMXBs split into the 2 individual systems.The colour of plot shows the fraction of time the simulation appears similar to an observed LMXB.

Rate Calculations

The rate calculations are done using the `rate_calc.py` python code. The final output of this process is the "rate_summary.dat" file which outputs the name of the LMXB, the mass - period combination of the

most optimistic rate, the most optimistic rate and the average rate as given by:

$$N^k_{\rm obs} = \Gamma^k_{\rm av} \frac{1}{\Delta A^k} \sum_m \tau^k_{ij} dA^k_{ij} \ . \tag{3}$$

I have included two example scripts that demonstrate how to generate the summarized rates and create a figure using the data:

- `rate_calc_example.py`: This runs through the steps on generating the summarized rate tables
- `create_rate_plot_with_labels.py`: This code generates a figure in donor mass - period plane where the colour shows the progenitor formation rate along with labels in our progenitor space gap.

While these two examples can get you the outputs used in the paper, I have included documentation/comments within the code and I recommend you skim through it to learn what else the `rate_calc.py` can do.

Gaps in Progenitor Space

To simplify the process in plotting simulated systems that don't result in observable LMXBs, I wrote the following code to hopefully streamline the process. Within `non_obs.py` contains code that helps read in and plot simulation data. An example of this is given by `non_obs_figs.py` that produces figures showing the evolution of a subset of binary simulations. The figures will show the evolution in mass ratio - period plane where the colour shows the mass transfer rate of the system. We also include the locations of the observed LMXBs.

Random Realizations

To test randomly producing our observed LMXBs I created a code that randomly select a simulated system that results in a persistent LMXB. This code is `random_rates.py` and requires determining which systems are persistent to begin with. To properly do this you need to run `find_pers_time.py` in the group directories similar to how `calc_duty_cycle.py` was run. This will output a file that contains the following information

- initial binary period
- initial donor mass
- total time spent as a persistent LMXB
- total age of the simulation
- age at start of mass transfer
- donor mass at start of mass transfer
- donor radius at start of mass transfer

- donor luminosity at start of mass transfer
- mass transfer rate at end if simulation

To calculate a random realization we need a code to randomly sample our simulated LMXBs, determine if the simulation is a persistent system then compare it to our observed sample. An example code of a pythron script doing this is included in `example_rand.py`. One caveat is that the random code requires that the user has generated a table of observed LMXBs and the calculated progenitor rate to use.