CTA200 RV Project

Fergus Horrobin

In this project you will use the publically availale NBody dynamics code `REBOUND` to simulate the radial velocity (RV) curve we might find when observing an exoplanetary system. The radial velocity curve is a measure of the relative motion of the star due to gravitational interactions with orbiting planets. For some background on the radial velocity method, this review provides some good background (see section 2 in particular).

1 Using `REBOUND`

The first step of this project is to get familiar with simulating planetary systems using the publically available NBody simulation code `REBOUND`. You can see the code at `https://github.com/hannorein/rebound`. If you are using the CITA machines, `REBOUND` is already installed, just load the Python module and you can import it. If you are using your own machine, install it using pip.

1. Go to the documentation for `REBOUND`. Find the Quick Start Guide and follow the instructions for running your first simulation.
2. By default the units are set such that $G = 1$ (a common choice by people in the planetary NBody and planetary disk community, though not one I am a huge fan of). Search the documentation to find how to set the units to ('yr', 'AU', 'Msun'). Make sure you set the new units before adding any particles to the simulation.
3. Modify the setup from the example in the quick start guide so that you are simulating two planets with the following parameters: $(m_1, a_1, e_1) = (10^{-3}M_\odot, 5.2AU, 0.05)$ and $(m_1, a_1, e_1) = (3 \times 10^{-4}M_\odot, 9.6AU, 0.05)$ around a $1M_\odot$ star. Run the simulation to a time of 500 years.
4. Plot the orbital trajectories of both planets on the same plot. Also make a plot of the semi-major axis and eccentricities of each planet as a function of time.

2 Making a Synthetic RV Curve and Understanding RV

In this section you will use `REBOUND` to simulate what an RV curve might look like for a certain planet configuration.

1. Make a simulation with a single planet of mass $10M_\oplus$ orbiting a $0.1M_\odot$ star on a circular orbit at 0.1 AU. Run the simulation for 30 orbital periods of the planet.
2. Plot the radial velocity of the star as a function of time (Hint: choose the direction to the observer such that it lies either along the x or y axis, think about what this means for the radial velocity). Make sure you have at least 10 data point per orbit.

3. Separately vary the mass of the star, the mass of the planet and the semi major axis. How does the radial velocity curve depend on each of these?

4. Repeat the simulation with $e = 0.3$. Make another plot of the RV curve. What changes?

5. Re-plot the RV curve from the previous question but this time make the units for time days and the unit for radial velocity cm/s (don't re-run the simulation with new units, just convert the units from the simulation you ran before).

3 Fitting RV data using `REBOUDND` and `emcee`

In this part, you will build on the synthetic RV modelling you did already to try to fit the orbital parameters of real RV data. To do this, you will sample different sets of orbital parameters using the MCMC sampler `emcee` and will compare the synthetic RV with the real data for each realization. You will use the data and methods from Silburt & Rein (2017) (hereafter SR2017) and try to reproduce their fits. This is a system that is believed to have two planets and that has a pretty good and easily available RV dataset.

3.1 Reading the data

1. There is a file called data.txt in the project folder for the RV data. Download the data file and load the data into 3 numpy arrays for the day, radial velocity and error.

2. Plot the radial velocity with errorbars as a function of the day.

3.2 Testing your synthetic RV

Using the parameters found in RS2017, run a rebound simulation which simulates the RV curve. Plot this curve over the data and make sure the curve and data agree.

3.3 Learning about emcee

Go to the documentation for emcee specifically the page on fitting a line `https://emcee.readthedocs.io/en/stable/tutorials/line/`.

Reproduce the results of this example to make sure you understand what emcee is doing.

3.4 Defining the model

We will fit the following likelihood function for our data:

$$\mathcal{L} \sim \Sigma_i \exp\left(\frac{-((v_{sim_i} + \gamma) - v_{obs_i})^2}{\sigma_{rv_i}^2 + \sigma_j^2}\right) \tag{1}$$

Where v_{sim} is the simulated radial velocity curve, γ represents the motion of the entire system relative to the Earth, v_{obs} is the observed radial velocity value from the data set, σ_{rv} is the error from the RV dataset and σ_j is a number to encode scatter in the data not captured by the observed uncertainty.

To actually fit this, we need to introduce the parameters needed for v_{sim}, these will be: $m_1 \sin i$ and $m_2 \sin i$ (the planet minimal masses), a_1 and a_2 (planet semi major axes), e_1 and e_2 (eccentricities), ω_1, ω_2 (argument of pariapsis), M_1 and M_2 (mean anomoly) (you may have to look up what some of these mean, you could for instance see the Wikipedia page on orbital parameters). Because there is a singularity for $e = 0$, we will instead fit the parameters h_1 and h_2 where $h = e \sin \omega$ and k_1 and k_2 where $k = e \cos \omega$

We will also have to define priors for the model, look at the paper linked at the start of this section (RS2017) and implement the same uniform priors used in that work.

3.5 Fitting the model

Here we will actually implement the model in a way that emcee can use it. Since the likelihood as written above has a large range of values, it is common to instead maximize the $\log \mathcal{L}$.

1. Implement a function for $\log \mathcal{L}$, this function should take the current sample drawn from emcee which will be a vector of $\theta = [m_1 \sin i, m_2 \sin i, a_1, a_2, M_1, M_2, h_1, h_2, k_1, k_2, \gamma, \sigma_j]$. It should then use $[m_1 \sin i, m_2 \sin i, a_1, a_2, M_1, M_2, h_1, h_2, k_1, k_2]$ to compute a synthetic RV curve for the system which should return the simulated RV, sampled at the same time as the observed RV. Then use the simulated RV as well as γ and σ_j to compute the $\log \mathcal{L}$ and return this value.

2. Implement another function that takes in the same vector θ as before and returns the log prior (either 0 if all the parameters are in the allowed range, otherwise $-\infty$).

3. Implement a log probability function (the probability is the prior * the likelihood so log probability is just log prior + log likelihood).

4. Create 20 walkers initialized near the values found in RS2017 (give each walker a different random perturbation away from the values found in RS2017 by adding a normally distributed random number scaled to about 5% the value).

5. Run the walkers for 6000 steps. Remove the first 1000 steps as burn-in. Combine all of the walkers for the remaining steps. Find the mean of each of the parameters (the mean values from all of the wealkers for all of the steps). Are the values similar to those which RS2017 found?

6. Think about how you can estimate the errorbars on each of the values using the data in the MCMC chain. Cite the values with errors in your writeup.

7. Plot a synthetic RV curve using the mean values you found for each parameter as well as the raw data. Does the fit look good?

8. Choose 50 random samples from the MCMC chain and plot these on the same plot as thin lines with alpha=0.3.

9. Use the corner.py package to make a corner plot of all of the parameters.