SURP 2021: Tattooine planets introductory project

Kanah Smith

1. PRELIMINARIES

We'll start out with the necessities. If you haven't already, first:

1. Install a distribution of python including numpy, scipy and matplotlib. Anaconda (https://www.anaconda.com/products/individual) is probably simplest

2. Install ipython/Jupyter notebooks (slow, but useful for sharing work)

3. Find a text editor you like. Popular options include gedit (for the minimalists), Emacs (for the hardcore), Sublimetext (a bit expensive, but has a free trial) and Visual Studio Code (lets you run terminal windows and ipython notebooks right next to your code).

4. Install REBOUND (https://rebound.readthedocs.io/en/latest/python_quickstart.html)

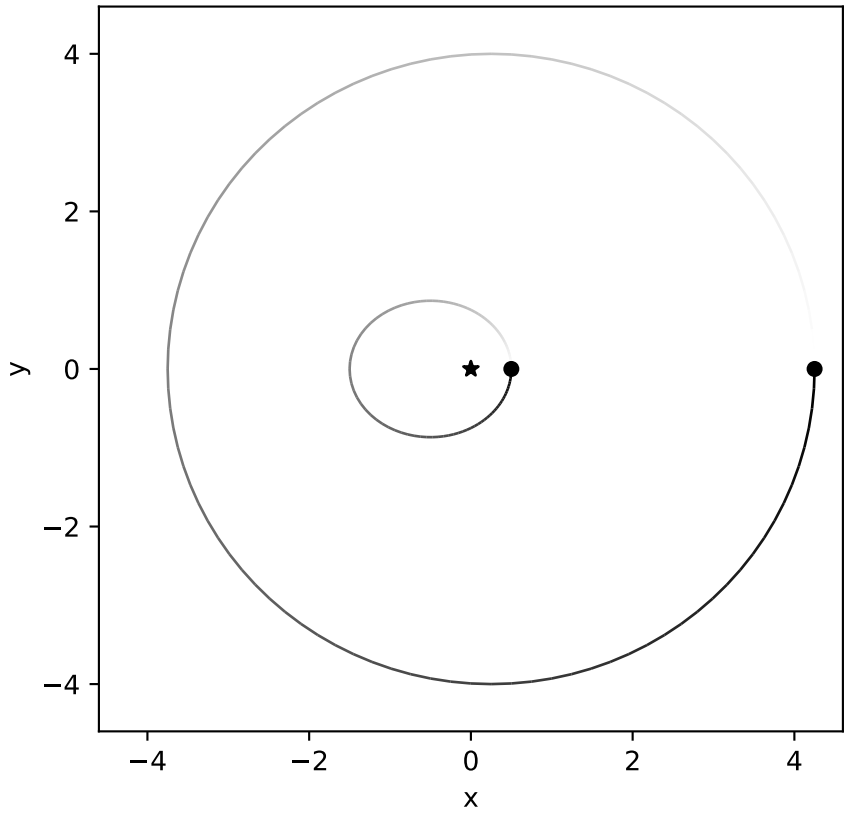
2. INTRODUCTION TO REBOUND

1. Become familiar with REBOUND by reading through the example files in the REBOUND documentation.

2. The code creates a simulation over multiple barycentric orbits for a system with a binary star pair and an orbiting test particle planet while positions for the stars and the planet are collected at user specified intervals. (The points were collected properly using a Simulation Archive but the computer/verion of REBOUND available at the time could not configure the .bin file correctly. **working to fix this). The simulation will stop if the the planet flies beyond 4 times the binary semi-major axis.

3. This code creates another simulation over several orbits and compares a manual plotting of orbits using parameterization and matplotlib versus the RE-

BOUND built in function OrbitPlot. Below is the image created using Orbit-



Plot:

My manually made plot compared to REBOUNDs OrbitPlot differs as it can show me the orbit of the first binary star, which in OrbitPlot serves as the center and therefore everything orbits around that central body.

What happens if you set $a_p/a_b = 3$, and why?: If I set the ration of the semi-major axes $a_p/a_b = 3$ to 3, changing a_p from 4 to 3, has no affect on the binary pair since the planet is a test particle planet with no mass and therfore no gravitational influence.

3. CLASSIC RESULTS

We'll now reproduce the stability criterion found by Holman & Wiegert (1999) for 'P-type' orbits. For reference, they found a critical

$$\begin{aligned} \frac{a_c}{a_b} &= 1.6 + 5.1e - 2.22e^2 + 4.12\mu - 4.27e\mu - 5.09\mu^2 + 4.61e^2\mu^2, \\ &\approx 2.278 + 3.824e - 1.71e^2. \end{aligned} \tag{1}$$

the latter approximate equality following from the authors' observation that a_c is (roughly) independent of the binary mass ratio.

1. This code that was modified from section 2 creates function, which when called creates a simulation with a binary pair and orbiting planet. The functions argument, which is a tuple of varying values of binary eccentricity and planetary semi-major axis, is used to find those values in the simulation at which the planet is ejected. The function should return a 2D array of the mean survival times of those tuple values.

   rebound.InterruptiblePool.map is used to do parallel data processing on those varying tuple values that have been added to a list referred to as tup_{list}.

   ** The function still needs to be fixed to create the correct 2D array, but for now only returns 7 survival times instead of the expected 25 from the 5 tuples evaluated. ** A plot could not be made yet as the 2D array produced earlier still must be fixed to output 25 values

REFERENCES

Holman, M. J., & Wiegert, P. A. 1999,
AJ, 117, 621, doi: 10.1086/300695