Draft version January 4, 2021
Typeset using LaTeX default style in AASTeX63

Testing the Stability of Planetary Systems with Two Jupiters

Isaiah I. Tristan[1]

[1] The University of Colorado Boulder
2000 Colorado Ave
Boulder, CO 80305, USA

(Received December 1, 2020; Revised December 10, 2020)

Submitted to Planetary Origins

ABSTRACT

It is thought that our planetary system may have developed instabilities after formation if there were two Jupiter-like planets in the mix, through large orbital wobbling/eccentric orbits or ejections. To investigate this problem, I created an orbital motions simulator and placed a second Jupiter (Tupiter) at different locations to see the effect caused by it. I also studied the limits of the perturbations caused between the two giants if they were alone and tested how two Jupiter-like planets would affect randomly generated systems, as opposed to only studying our Solar System. I found that even on short timescales (< 500 years), Tupiter could cause instabilities and found random system stability is more affected by the number of planets it has and their placement in general, rather than just having two Jupiter sized planets around.

Keywords: planetary systems – numerical simulations

1. INTRODUCTION

Protoplanetary disks may have kept planets in certain spots after formation, and that once the disks dissipated, large instabilities could form in the system, causing orbital perturbations leading to eccentric orbits or ejections. Theories like the Nice model also support the thought of instabilities leading to post-disk migrations (though perhaps in more dramatic ways) (Gomes et al. 2005; Tsiganis et al. 2005; Morbidelli et al. 2005). With this in mind, I asked, **”What if there were two Jupiter-like planets instead of one? Would our system be completely unstable?”** I also found that this is a question that has been studied before in literature. Rasio & Ford (1996) found that two Jupiter planets may indeed cause instabilities and ejections when placed in a planetary system, Ford et al. (2001) found that if only one is ejected, the remaining may have a smaller, more eccentric orbit, and Morbidelli (2018) found that in any planetary system, the more planets there are at the beginning, the more instabilities there are.

Given this background, let's say there was a second Jupiter (Tupiter). Some of the literature looks at small perturbations that take place over hundreds of thousands of years, but with the tools available to me, I needed something that could be computed fast. So, I decided to focus on what kind of short term instabilities (within 500 years) Tupiter could happen, hoping to see the same type of instabilities as mentioned above. To tackle this, I planned to make a code that could model 3D stellar system orbital mechanics and test the interactions between Jupiter, Tupiter, out Solar System planets, and arrays of randomly generated planets. I believe this question is worth studying as it deals with creating methods to tackle N-body dynamics and ultimately matters in planetary system habitability (e.g. if two large planets make systems extremely unstable, how many systems are unstable from the start?).

In section 2, I talk about the methods used and codes developed. In section 3, I write up the results of the work. In section 4, I draw conclusions and return results to the context of the original questions, as well as discuss future improvements that could be made. Finally, in section 5, I respond to the questions received after the class presentation.

Corresponding author: Isaiah I. Tristan
Isaiah.Tristan@colorado.edu

2. METHODS

2.1. *Physics and Math of the Simulation*

In order to study this these short term instabilities, I first had to develop a program that could simulate solar systems and planetary motion, which I called the Planetary System 3D Realistic Orbital Motions Simulator (PS3-ROMS)[1]. In deciding on what parameters to look at and what physics to look at, I chose to focus only on planetary mechanics. That is, this system would only have one star & N planets and would be governed only by the forces of gravity between the objects,

$$F_{G,\text{total}} = \sum_{i=1}^{N} \frac{Gm_0 m_i}{r_{0-i}^2} \tag{1}$$

where F_G is the gravitational force, G is the gravitational constant, m_0 is the mass of the object, m_i is the mass of the other object(s), and r_{0-i} is the distance between the two objects. The star was held at a Solar mass and at the origin for this project, so that we could see what would happen in a system with a star like our own. With this set up, we are only testing Solar System-like systems in the stellar frame of motion. We are also neglecting dust and gas, with any physics that would accompany those, as well as other small bodies (asteroids, comets) and any relativistic effects that would occur for planets that move close to the Sun.

I then had to decide on a time stepping method to use to propagate the system. While this might be one of the more boring parts of a physics project, it can be incredibly important, as a lacking method will lead to nonphysical results. First, I added in the simplest method possible, the Euler method, where positions and velocities of objects are updated using

$$\begin{aligned} p_{n+1} &= p_n + v_n \delta t \\ v_{n+1} &= v_n + a_n \delta t \end{aligned} \tag{2}$$

where p is position, v is velocity, a is acceleration, δt is the time step value, and $n/n+1$ denote the current/future steps. However, it is well known that the Euler method is not good at maintaining energy conservation (Garcia 2000), so a better method would be necessary to run any tests.

The Euler-Cromer method instead switches the order in which you update velocity and position.

$$\begin{aligned} v_{n+1} &= v_n + a_n \delta t \\ p_{n+1} &= p_n + v_n \delta t \end{aligned} \tag{3}$$

While the difference is simple and does not require the code to do any extra work comparatively, the difference in accuracy is significant, as the Euler-Cromer method adds a predictive component to the object movement. This is the minimum necessary to run orbital models (Garcia 2000).

In order to reduce the code's run-time a bit, I used "Solar System Units", where length is kept in AU, masses in $M_\odot$, and time in years. This causes calculated values to be within 10^{-5} to 10^{5}, rather than the 10^{20}+ ranges of cgs units, which gives the computer a little break on the length of the calculated numbers.

It should be noted that PS3-ROMS also has a rudimentary way of dealing with collisions. For each step, it calculates is any planets are within $2R_{\text{Jupiter}}$, and if so, it removes the smaller planet and adds the mass to the larger one. This works well for sucking up smaller planets, but can cause anomalies when one of the larger planets suddenly disappears.

2.2. *Stellar System Setups*

Next was setting up the planetary system itself. I added two modes to PS3-ROMS, Solar System and Random System. In Solar System mode, the eight planets in our solar system were used placed in XYZ-space around the Sun using their known semi-major axis, inclination angles, and masses (see Table 1).

In Random System mode, the user can specify the number of random planets to make. The program then generates distances from the Sun from 1 to 30 AU, masses between 1 and 17 Earth masses, and inclinations between -3 and 3 degrees. It then adds Jupiter in its normal place.

In both modes, Tupiter is then added in at a user-defined distance with the same mass (technically a bit larger, for code purposes) and inclination of 0 deg. Allowing the user to set this makes it easy to test for when the Jupiter-Tupiter

[1] Source code is available at https://github.com/isatri/isatri.github.io/blob/master/PS3ROMS.py

Name	Mass	r	incl.
	($M_\odot$)	(AU)	(deg)
Mercury	1.66e-07	0.4	7.0
Venus	2.44e-06	0.7	3.4
Earth	3.00e-06	1.0	0.0
Mars	3.22e-07	1.5	1.84
Jupiter	9.54e-04	5.2	1.3
Saturn	2.85e-04	9.6	2.5
Uranus	4.36e-05	19.2	0.77
Neptune	5.14e-05	30.0	1.77

Table 1: Known data values for the Solar System's planets (from https://nssdc.gsfc.nasa.gov/planetary/factsheet/).

distance causes instabilities (in the Solar System model, at least). All objects are started at the Y=0 axis and then given a starting orbital velocity of

$$V_{\rm orb} = \sqrt{\frac{GM_\odot}{r}} \tag{4}$$

(where r is the starting distance from the Sun) in the positive Y direction.

Since we are only looking at short term instabilities, the code is set to run for 500 yrs using 0.01 yr time steps as the default values. However, these can be changed by the user easily, if needed. With this setup, PS3-ROMS usually completes its runs in about 5-20 seconds depending on the number of planets. Running for 200,000 yrs with 0.01 yr steps (that is, testing for system stability) runs at about 20 minutes. The positions and velocities of all objects are saved for each time step in arrays to be plotted/tested afterwards.

2.3. *Parameter Testing*

The saved positions and velocities were then used to check semi-major axis, eccentricities, and energy calculations. Semi-major axis were calculated simply as the average distance from the sun for nearly circular, while eccentricities and total energy were calculated (Carroll & Ostlie 2006) as

$$e = \frac{R_a - R_p}{R_a + R_p} \tag{5}$$

$$E_{\rm Total} = {\rm KE} - {\rm PE} = \frac{1}{2}mv^2 + \frac{GmM_\odot}{r} \tag{6}$$

where R_a is the aphelion distance and R_p is the perihelion distance.

These numerical calculations were mostly meant to compare with real values using the Solar System simulation, while testing Tupiter was designed to be more qualitatively described (stable, unstable, new orbits, and so on), depending mainly on the user-chosen values of distance (and number of extra planets for the Random Systems).

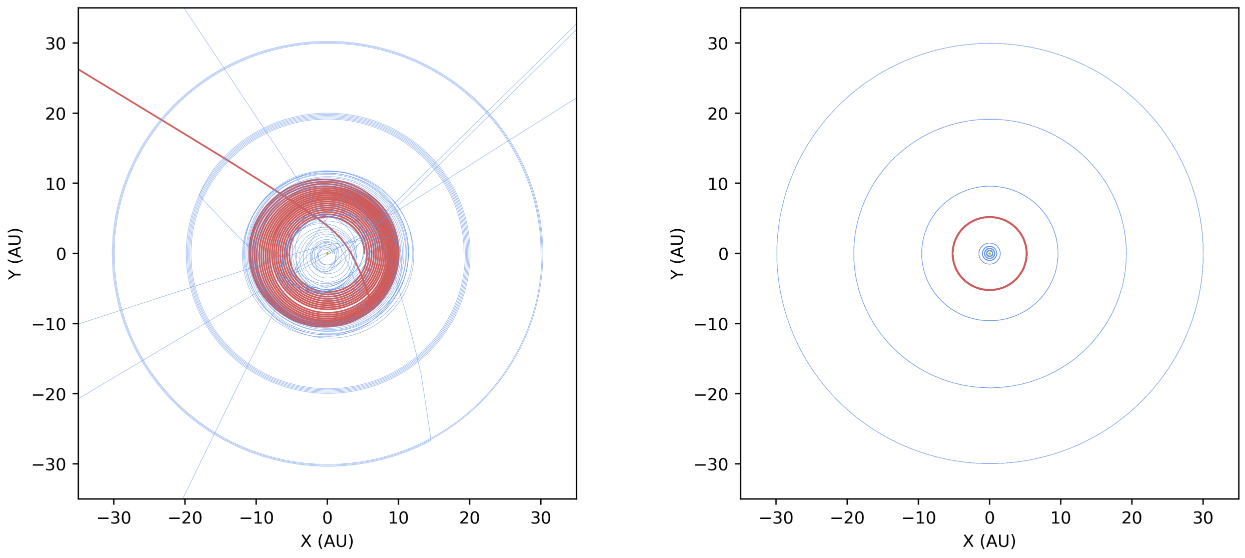
3. RESULTS

3.1. *PS3-ROMS Considerations*

First, I quickly tested the Euler method against the Euler-Cromer to show that the former was not enough to test these systems. In Figure 1, we can see that the Euler cannot produce orbits while the Euler-Cromer does so stably. In all following parts, we will only use results using the Euler-Cromer method.

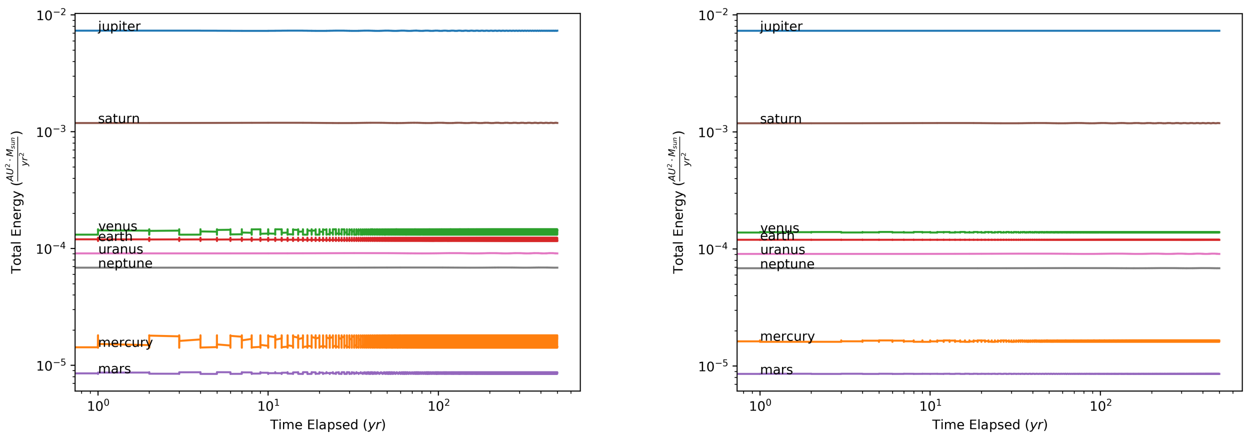
Also as expected, the Euler-Cromer method we use is still not perfect. In checking the conservation of energy of stable systems, there are small fluctuations that occur. None of these run away without very close encounters, so it still works reasonably well. In Figure 2, we can see that smaller time steps led to better energy tacking, but with the 0.01 yr time step codes running within 20 seconds, while the 0.001 yr steps running in minutes and close encounters breaking the codes either way, it made more sense to use the former.

3.2. *Simulation Outcomes: Solar System Testing*



(a) Euler Method (b) Euler-Cromer Method

Figure 1: I tested these two methods and found that only the second could handle the systems I planned to simulate. We can see that orbits are unstable using the Euler method, with some energy being lost, causing the planets to spiral inwards. Wacky gravitational pulls and orbit crossings caused many of the planets to be ejected by the end of the simulation. On the other hand, the Euler-Cromer method was consistent within reason. In these images, Jupiter is shown in red, while the other planets are shown in blue. In subsequent images, Tupiter will also be shown in red.



(a) 0.01 year time steps (b) 0.001 year time steps

Figure 2: For a perfect simulation, the energy of the system should stay constant in time. Here, it is shown that lower time steps give better results in keeping track of total energy. In general, for Euler-Cromer codes, lower time steps give better results, but for close encounters, the time steps required would make the code take a significant amount of time to run.

With tests using our own Solar System, things ran reasonably. In Figure 4, most values are shown to be close to the the true values, or at least within a magnitude. The eccentricity comparison is worse than the semi-major axis, so the cause of these changes is likely the lack of smaller objects that would give small perturbations to the planets.

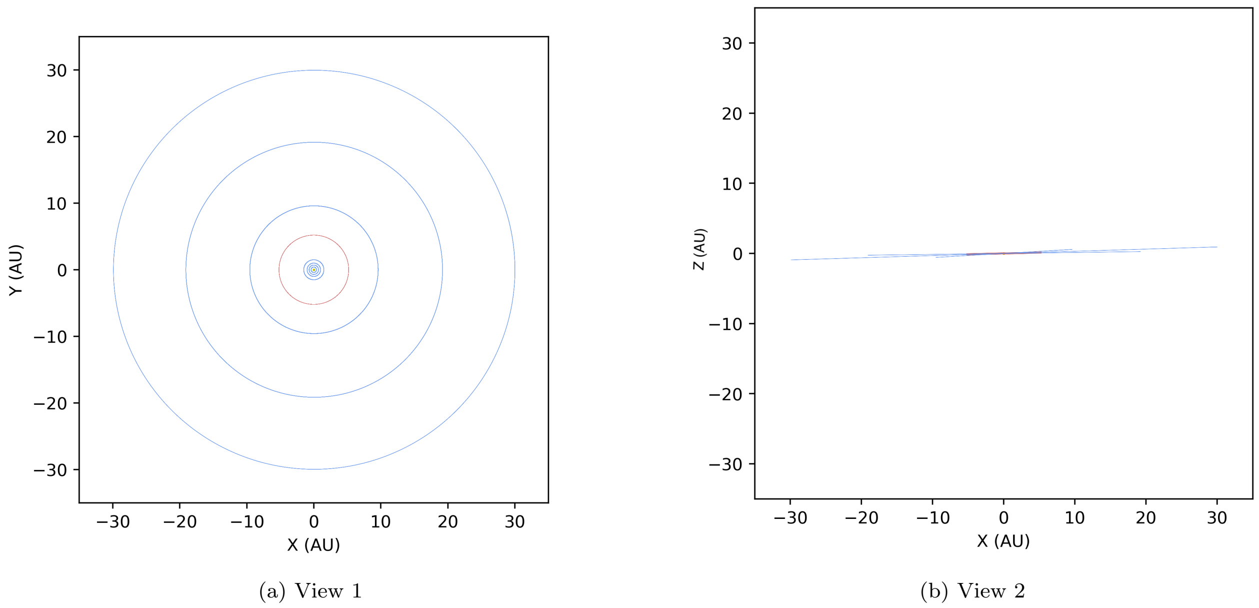


Figure 3: Orbits using the Solar System without Tupiter.

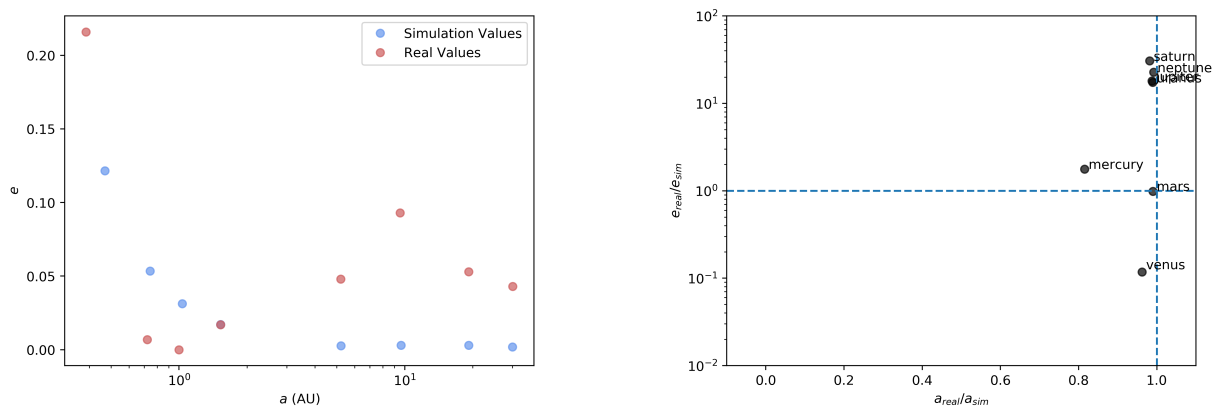
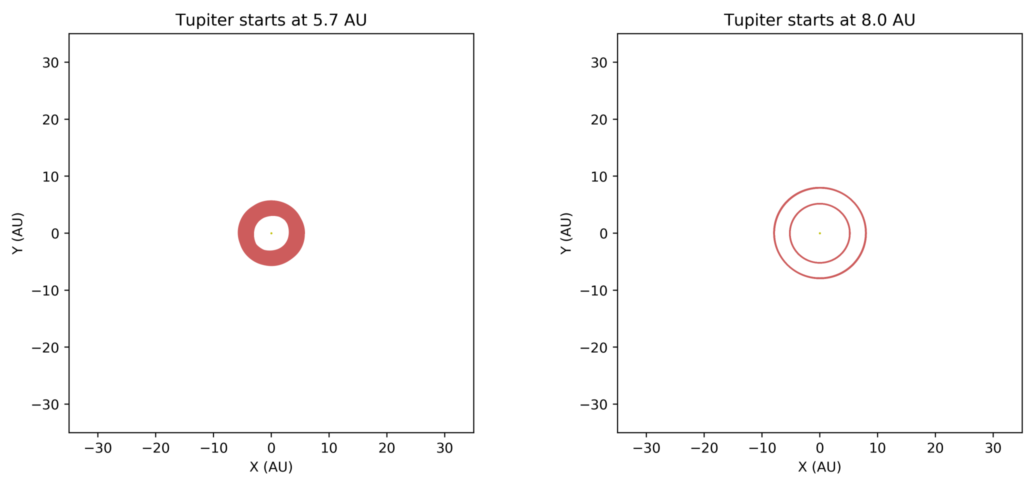


Figure 4: The simulation produces planets that are generally close to our own, but not exactly. The differences seen could be caused by the lack of asteroid belts, moons, comets, and resonant objects. The lack of relativistic treatment could also affect close-in planets like Mercury.

3.3. *Simulation Outcomes: Jupiter-Tupiter Testing*

When introducing Tupiter, I first tested where instabilities would be caused using only Jupiter and Tupiter. I found that if Tupiter starts far way and moves in, perturbations between the two start at around 8 AU, and the code would break (using 0.01 yr time steps) when less than 5.7 AU (see Figure 5). If I lowered the time steps to 0.001, they could get down to 5.5 and be kind of stable, though they entered a ”dancing” orbit (see Figure 6).

When I placed Tupiter in the Solar System simulations, it also did not cause too many problems as long as it was not near Jupiter (less than a difference of 0.3 AU) or Saturn (less than a difference of 1.5 AU). In both cases, Tupiter would instigate larger wobbles in the other planets orbits as it got closer. When the ”dancing” orbit was invoked, Jupiter and Tupiter entered the inner planet’s orbits, and either absorbed (shown as moving to the upper right on the graphs) or ejected the inner planets except for Mercury, which stayed in its orbit. When Tupiter was too close

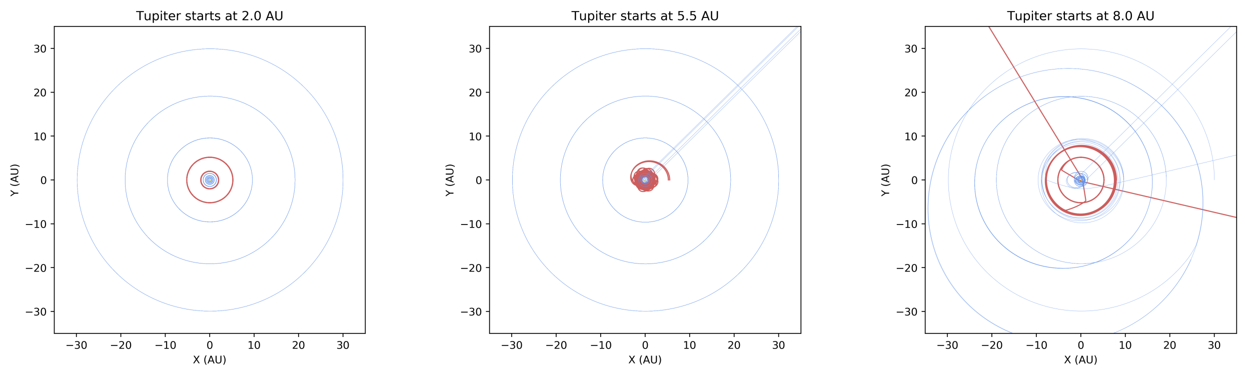


(a) Large orbital perturbations produced, but orbit fine

(b) Stable orbits

Figure 5: Systems with only Jupiter and Tupiter, used for sensitivity testing. On the left, Tupiter at 5.7 AU was the closest it could be placed using time steps of 0.01 before the orbits broke. After Tupiter was at 8 AU or further, the planets did not perturb each other much. If the time step was reduced, Tupiter could get as close as 5.5 AU (see next figure).

to Saturn, the perturbations caused the planets to get out of their orbits, dragging some of the other planets with them. In Figure 6 (c), Mars was dragged into a highly elliptical final orbit, and Uranus' and Neptune's orbits were also messed with. There was also some break in the code that caused Jupiter and Tupiter to fly out, but the other orbits seem to remain consistent.



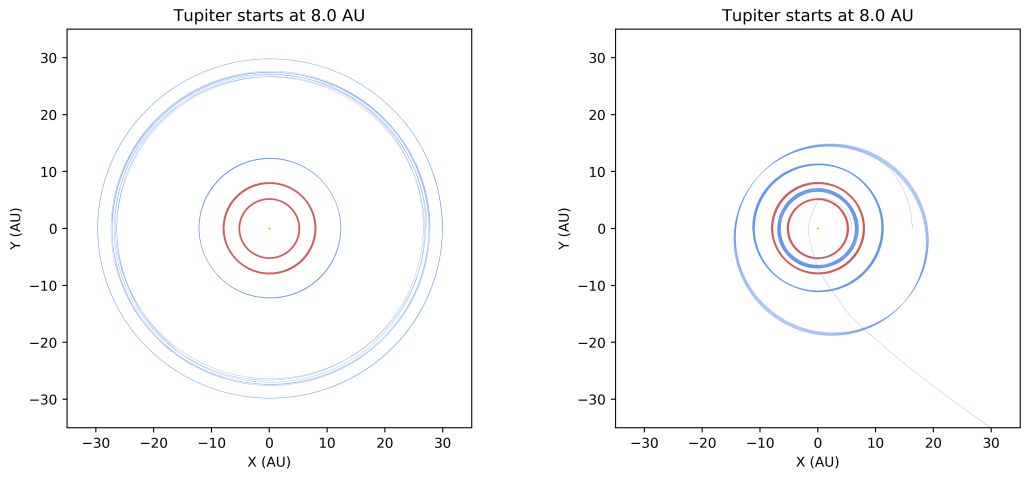
(a) Tupiter close to inner planets

(b) The "dancing" orbit using 0.001yr steps

(c) Tupiter close to Saturn

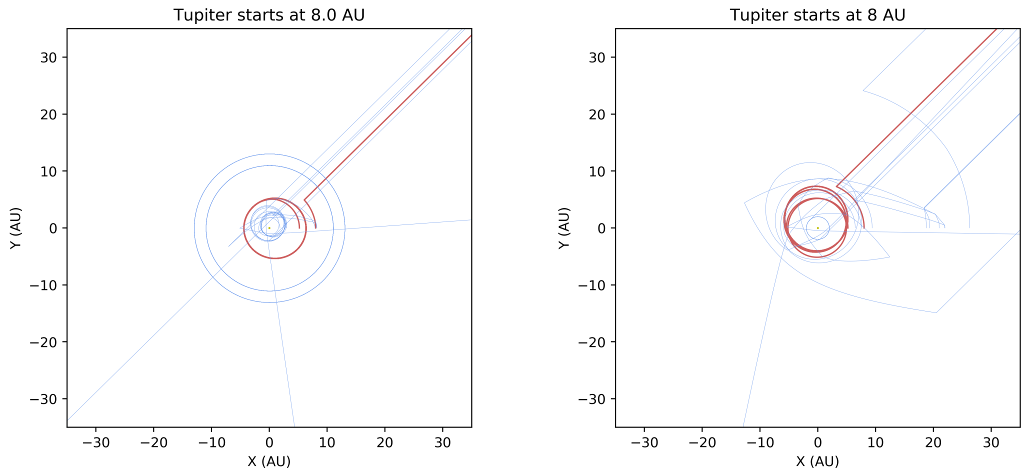
Figure 6: When Tupiter was placed in our Solar System, it did not cause too many problems except for when it was placed too close to Jupiter or Saturn.

When it came to the Random System simulations, I decided to keep Tupiter at a distance of 8 AU. From the previous tests, it was already shown that if Jupiter and Tupiter had giant perturbations, they would eject things in their path, so I decided to focus on if a random system would be stable if they were decently far apart. I found that for a low number of extra planets (4 or less), systems were mostly stable large wobbles in orbits being somewhat common. Most planets had a decent amount of space between them (see Figure 7 for examples). For systems with more than that, the rate of instabilities increased, with most systems of 7 planets or more having many ejections and anything over 10 planets breaking the code in some way or another (see Figure 8 for examples). In testing systems of 7 extra planets through 10 trials using 0.0005 yr time steps, none remained stable (3 had large code breaks). There did seem to be a dependence on the random planets' masses also, especially in the middle ground of 5-6 extra planets. There were more instabilities on average when all masses were higher, and small planets (close to 1 Earth mass) would get eaten by others easily without causing too many changes.



(a) 4 planets, test 1 (b) 4 planets, test 2

Figure 7: Examples of random systems with 4 extra planets. These were generally stable excpet for a few perturbed orbits and eccentricity changes, unless a planet spawned too close to one of the giants. If ejections happened, they usually did not break the code or drag others with it.



(a) 7 planets (b) 10 planets

Figure 8: Examples of random systems with large amounts of extra planets. These generally went haywire, either from lots of close interactions dragging things around or code breaks (from these interactions). Most of the time, there were at least one or two planets that survived, but they were not always guaranteed to be one of the two giants.

4. CONCLUSIONS AND DISCUSSION

From these results, we see that some of the expected outcomes were met, but some of the finer details need a more rigorous treatment. In general, the addition of Tupiter was able to cause instabilities in our Solar System within a short time (500 years). This was highly dependent on the distance between Tupiter and the outer planets, with a larger induced wobble effects and eccentric orbits seen on planets further out (seen with Jupiter and Saturn) than on inner planets (seen with Mars). If Jupiter and Tupiter were close enough to enter their "dancing" orbit, they passed through the area of the inner planets and ejected those. If they were too close, they would eject each other from the system, throw out other planets they passed by, and possibly break the code.

In random systems, instabilities were also mostly dependent on the distances between planets. When they were all spaced far enough, they would have circular orbits without problems. There seemed to be a small dependence on the average planetary size, with more massive planets resulting in more instabilities and smaller planets getting eaten without affecting the larger ones much, but this may be caused by the code not being accurate enough. The more planets there were, the more unstable the systems were, with 0 out of 10 trials of 7 extra planets remaining stable. This may be an artifact of the coding, as random planets could have spawned unrealistically close to each other, causing great increases in acceleration. If not, perhaps it is lucky that our system is so stable.

In both types of systems, Jupiter and Tupiter (or both) could be ejected, which sometimes resulted in the other have a smaller, more eccentric orbit. This effect could also happen with the random planets, not just the largest two.

In conclusion, this simple system has shown that two Jupiter-like planets may be detrimental to a system by causing instabilities or ejections, confirming the results of past literature, but ultimately the effects would be dependent on where Tupiter would form and what lies around them.

4.1. *Future Work Needed*

This project has allowed us to gain a few insights on what would happen in there were two Jupiter-like planets in a planetary system, but the results could be better. Given more time (or maybe less projects), there are a few changes I would like to implement that would make major differences.

On the smaller side:

1. I would like to implement at least 4th-order Runge-Kutta time stepping methods into the code, which are the industry standard for accurate N-body simulations.

2. In addition to upgrading the method, I would like to make it adaptive. an adaptive method checks the change of a parameter you select (like acceleration) and calculated the change that would happen between step n and $n+1$. If the change is above some set threshold (standard 3 to 5%), the time step is reduced and the change is checked again. The time step goes up again when it doesn't cause drastic differences. This ultimately reduces the time necessary to run codes. Using PS3-ROMS as an example, it would use time steps of 0.01 yr for the most part and use lower when two planets get too close and sped up significantly.

3. In using the Random System setting, I would like to add a check to ensure that planets do not start right next to each other (or inside each other). That is, if two planets start at 4 AU, then one would get pushed +0.2 AU out. In addition, rather than making the systems completely random, basing the planet masses on their starting AU would give better estimations of planetary systems (with the occasional Hot Jupiter placed at small AU).

For more intensive changes:

1. I would like to add some kind of better collision handling, even if the planets are just treated as colliding solid spheres (no shattering). Having one disappear breaks the simulation more than I would like, and is nonphysical. It was more of a necessity than anything to stop planets from passing through each other and gaining massive amounts of momentum.

2. Last, the results could be standardized more. For the Solar System tests, I would do this by creating a chart of Tupiter distances in some increments (0.01 AU) and run simulations for all of them. For Random Systems, I would probably decide on different configurations, keep them constant, and then test the grid of Tupiter distances.

3. While this borders on changing the project completely, having a faster code would enable other objects like comets and asteroid belts to be put in and allow for longer time periods, so it would be good to test if instabilities form on million-year timescales from orbits that the grid of 500 year simulations show as stable.

4. As long as we are going out of the scope of this project, I might as well add that it would be good to run a Nice model with Tupiter in it to see what would happen.

5. CLASSROOM Q&A

I would like to thank my fellow classmates for providing feedback at the end of my in-class presentation. Here, I have responded to each question sent, with the original text in bold and the answer in regular text after.

1. **If you did a sensitivity analysis to see what parameters are the most important, what would you be looking for? Would you be trying to determine which parameters eject Earth? What do you think the results of this would be?**
   From these tests, I believe the best sensitivity tests would be 'distance between planets' and 'number of planets', and I would want to know what causes the system as a whole to be unstable. I think you could focus on Earth, but given that two Jupiter-like planets in our own system may be unlikely, I'm not sure if I would focus the whole project on the Solar System. Here, I mainly thought of it as a starting point to check the random configurations against. The results I presented were that Tupiter didn't make too much of a difference as long as it was not too close to Jupiter or Saturn.

2. **How does your time-stepping method compare to off-the-shelf N-body code in computational time and accuracy?**
   Assuming off-the-shelf is something like a NASA-grade N-body simulation or long-term open-source collaboration like Rebound, this is not as good. This was made for a class project with limited time and in Python. High-computation codes, even the ones you use with Python, are usually made with C or Cython, both of which run faster by default. There's some kind of argument for making your own codes versus using black-box ones on the internet though.

3. **How large of a role does the integration algorithm play in your results?**
   In general, the integration algorithm could play a huge role in the results! Like shown in the presentation, the Euler method could not even simulate an orbit. With the Euler-Cromer method, we can still see that choice of time-steps leads to different outcomes. With a better integration algorithm, we would not have as much code-breaking changes in simulations.

4. **Why model with super earths near Jupiter if this is compositionally unlikely? Is it just a size thing?**
   Some believe that Super-Earths/Mini-Neptunes might be the most common type of planets (Knutson et al. (2014) and references within) and it's unusual that our system does not have them. This drove my main reason, but given that the smaller planets tend to not have a large say in the outcomes, it worked out pretty well. The radii were chosen randomly between 1 and 30 AU for code simplicity.

5. **Is it possible to make a second Jupiter 5 AU when the original Jupiter is already forming? Is there enough material? Asking because if not that seems that the necessary migration of the second Jupiter to that point would lead to all sorts of instabilities and changes in the system.**
   I don't think so for our solar system (since Jupiter is about 70% of the planet's combined mass), but in other solar systems there could be the possibility of more material being centralized in a certain area. In a disk where two Jupiters were forming, if they are both moving in, it might not be too destructive (to each other, at least) which would allow for one of our tested system that only have the two Jupiters.

6. **Do you think adding the resonances to the initial conditions would change the results drastically?**
   For these short-term (500 yr) instabilities? Probably not, but for longer term integration, definitely. At its core, resonances do not have much 'chaos' in them. Barring interactions with other things, there is a consistent force happening throughout time, rather than having small differences from orbit to orbit that could lead to instabilities.

REFERENCES

Carroll, B. W. & Ostlie, D. A. 2006, An introduction to modern astrophysics and cosmology / B. W. Carroll and D. A. Ostlie. 2nd edition. San Francisco: Pearson, Addison-Wesley, ISBN 0-8053-0402-9. 2007, XVI+1278+A32+I31 pp.

Ford, E. B., Havlickova, M., & Rasio, F. A. 2001, Icarus, 150, 303. doi:10.1006/icar.2001.6588

Garcia, A. L. 2000, Numerical methods for physics / Alejandro L. Garcia. Upper Saddle River, N.J. : Prentice Hall, c2000.

Gomes, R., Levison, H. F., Tsiganis, K., et al. 2005, Nature, 435, 466. doi:10.1038/nature03676

Knutson, H. A., Dragomir, D., Kreidberg, L., et al. 2014, ApJ, 794, 155. doi:10.1088/0004-637X/794/2/155

Morbidelli, A., Levison, H. F., Tsiganis, K., et al. 2005, Nature, 435, 462. doi:10.1038/nature03540

Morbidelli, A. 2018, Handbook of Exoplanets, 145. doi:10.1007/978-3-319-55333-7145

Rasio, F. A. & Ford, E. B. 1996, Science, 274, 954. doi:10.1126/science.274.5289.954

Tsiganis, K., Gomes, R., Morbidelli, A., et al. 2005, Nature, 435, 459. doi:10.1038/nature03539

Note: All reference formatting is taken from the SAO/NASA ADS Service.