Computational Astrophysics Exam Project

Long Term Particle and Planet Orbits

Jakob H. Schauser, pwn274

January 2024

Contents

1 Introduction

In this project we will be looking at the so-called Kirkwood Gaps and the different computer-science hoops I had to jump through to make simulating their dynamics feasible.

2 Numba

2.1 Implementation

Even though the use of Python classes during the course have seemed afwully smart, I have never truly gotten my head around Object Oriented programming. It seems like Numba feels the same way, so, for both of our sake, I chose to rewrite in a (more) functional way.

Taking advantage of the compilation to C, the final implementation consisted of a bunch of functions looking like this:

```
@njit
def fn(x : ndarray) -> float:
    ...
```

Simply throwing decorators at functions is (unfortunately) not enough. Special attention must still be taken to the implementation. Firstly, even though for-loops are sped up manyfold, vectorized code still help the compiler optimize the python code. Secondly, while type declaration is not a requirement as it is in other python-to-`C`-compilers (cython, for example), it can be very helpful in debugging as `C` is very strictly typed.

Also using `prange` allows for easy parallelization, but (exept for when calculating the gravitational force) I deemed implementing this outside of the scope of a 72hour exam so I ended up opting out of this potential speed boost.

2.2 Speedup

For completeness sake, I have made a quantified, albeit unprofessional measure of the speedup using the increadibly smart iPython function `%timeit`. For a simulation of 500 particles over 1e5 steps on my laptop:

No Numba: 5min 6s ± 18.4 s per loop (mean ± std. dev. of 7 runs, 1 loop each)

With Numba: 7.64 s ± 412 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

3 Symplectic Integration

For all simulations, having the underlying science in mind can help implementation immensely. A prime example of this is *symplectic integration*. As is also explained in [FOREST RUTH], keeping the Hamiltonian constant is a canonical way of thinking in physics, as this equates conservation of energy. Where other integration schemes might have an arbitrarily small error per time step, there is no guarantee for the accuracy of the physical simulation over longer time scales as the variables (position for example) might accumulate error, therefore being less stable.

3.1 Theory

An example of a symplectic integration scheme is the leapfrog algorithm as we implemented in the course. But simply being symplectic does not equate YYY quality. There are some things to have in mind: Firstly, you can easily have symplectic algorithms that are only precise in $\mathcal{O}(dt)$. This is why for this project a $\mathcal{O}(dt^4)$ algorithm was implemented. Secondly, symplectic integrators approximate solutions on a closed curve in (q,p) or((x,v) phase-space depending on your formalism), meaning symplectic integrator will satisfy differential equations where the total energy is conserved; this means, that if we had drag in the solar system, it would not be trivial to utilize symplectic YY.

There are multiple different implementations of getting taking time-reversible integrations steps, but general form we will follow are as follows[1]:

$$\boldsymbol{p}' = p + V_x\left(x'\right)t, \quad x = x' + A_p(p)t \tag{1}$$

Where V and A is the potential and kinentic energy in the canonical ensemble as it is called in analytical mechanics.

Exactly like the leapfrog method, this t-step can be split up into multiple parts for negating lower order error. Since the v- and x-updates can be seen as a kick and drift respectively stringing together multiple kicks and drifts, each with their own coefficient in front can lead to more accurate solvers. As any symplectic step is simply a single dt step, the sum of the coefficients must equates exactly one, other than this, there are multiple solutions. Now all the troubles comes from finding the c_i and d_i's. Luckily people much smarter than us has been working in this field, and I will happily stand on their shoulders. I have chosen the coefficients found by Yoshida.
Using

$$w_0 = -\frac{\sqrt[3]{2}}{2-\sqrt[3]{2}} \quad \text{and} \quad w_1 = \frac{1}{2-\sqrt[3]{2}}$$

You can define

$$\begin{aligned} c_1 = c_4 &= \frac{w_1}{2} \\ c_2 = c_3 &= \frac{w_0 + w_1}{2} \\ d_1 = d_3 &= w_1 \\ d_2 &= w_0 \end{aligned}$$

The implementation was quite straight forward, but attention had to be made for the order in which the different parameters where updated. To summarize, it the point is to iteratively do the following:

$$\begin{aligned} x_i &= x_{i-1} + c_i \cdot v_i \cdot \Delta t \\ v_i &= v_{i-1} + d_i \cdot a(x_i) \cdot \Delta t \end{aligned}$$

4 Reflexive Time Steps and Integration Scheme Comparisons

As mentioned in class, a symplectic integration algorithm looses it's symplecticity[2] when variable time steps are used. For this reason, I have had a fixed dt during all of my simulations, but for testing I have implemented the 'reflexive time step' scheme. For a comparison of the effects of this implementation see Section A in the Appendix.

[1] Taken directly from[FOREST RUTH]
[2] This is a real word!

5 Kirkwood Gaps

5.1 Theory

Orbital Resonance is, apart from being an excellent New Age band name, one of those emergent phenomena that makes perfectly easy to rationalize about but is non-trivial to calculate. The gist is that when the periods of orbiting bodies are rational fractions of the period of a much larger body, it would always "see" the heavy in the same place in its orbit. This would mean, that it is always pulled in the same direction when in the same place. The periodicity will therefore over time pull the small body out of its previously regular orbit.

Assuming circular orbits, in our system of units, we have from Kepler's third law:

$$r = \left(\frac{P^2}{4 * \pi^2} \right)^{1/3} \tag{2}$$

As I will be looking at the Kirkwood gaps surrounding Jupiter, we are expecting the bodies to migrate away from

$$r = \left(\frac{(P_{Jupiter} \cdot \mathrm{f})^2}{4 * \pi^2} \right)^{1/3} \tag{3}$$

where f is some nice, regular fraction like $1/2, 1/3, 2/5, \ldots$

5.2 Implementation

I have opted to dust/asteroids that will form the gaps are non-self-interacting - allowing for a massive reduction in needed computations. Secondly, for simplicity sake, I started looking at completely circular orbits, only including the most influential planet (Jupiter[3]) and the most influential star (the Sun[4]). As I set out to recreate the results in YYY[A record of planet migration in the main asteroid belt], I started 400 asteroids at regular intervals between 2 AU and 4 AU.

5.3 Results

After applying all the principles above, we are finally ready to reap our rewards. This gives us Figure 1, which I find so cool that it deserves its own page (please don't count it towards my page limit). On it we can clearly see how, after a long time, the previously regularly spaced asteroids have migrated from the theoretically predicted bands:

[3]The heaviest planet in the solar system

[4]The closest star to the center of mass of our solar system

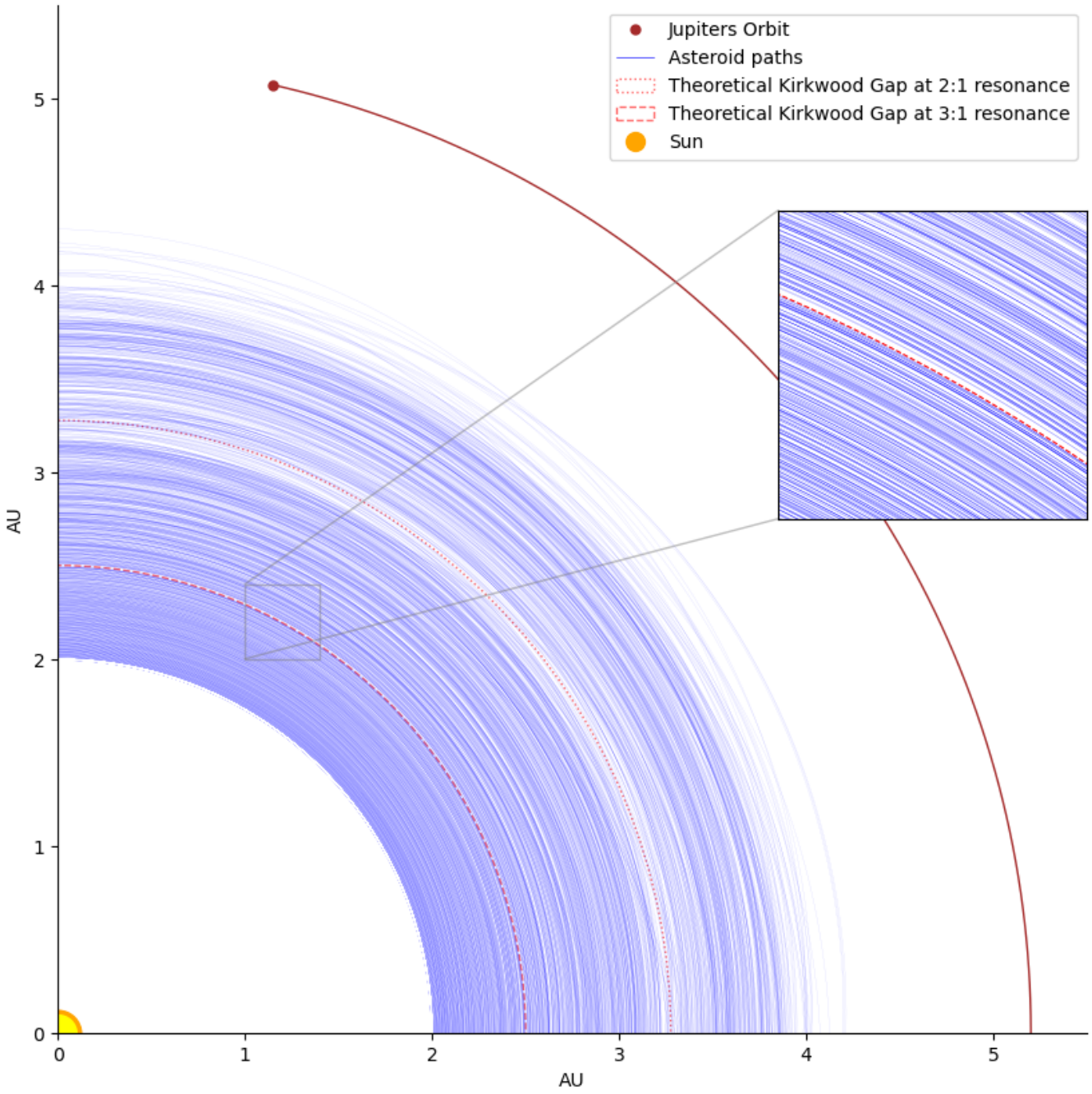
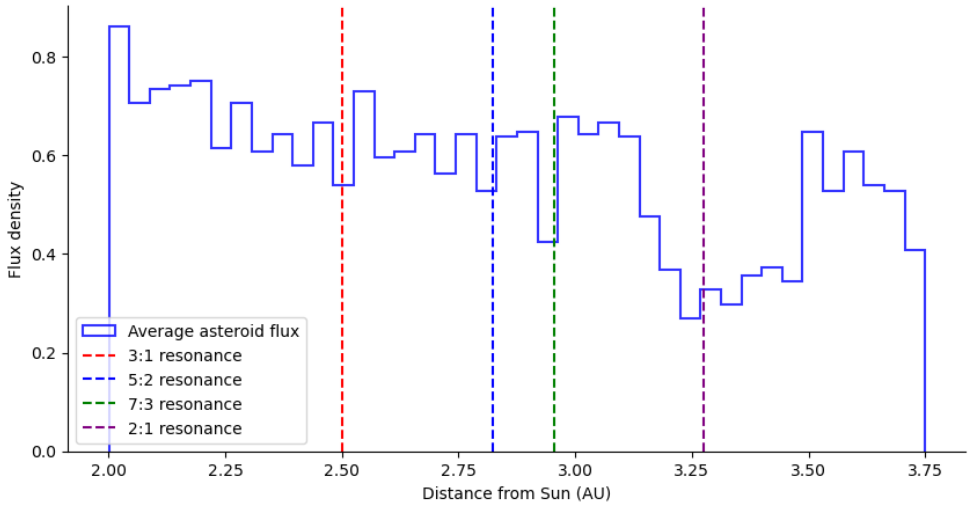


Figure 1: The money plot

As previously mentioned, my first idea was to recreate the bar-chart (figure 1) in YYY. This required a way of quantifying how many particles were moving through a certain band over time. The results can be seen here:



This is perfect!

The 2:1 band is clearly visible and we also refind the same gaps as are clear in the YYY paper, at around the same prominences. Does not look exactly like the one in the paper. This is due to multiple tings, mainly that I have not added Saturn to my simulation. For the distribution with Saturn added see YYY

5.4 Comparison of Symplectic integrators

For this analysis to be viable, I needed a way to solve the differential equation that was long-term accurate. This is why the fourth order was implemented. To quantify the improvement, I let the simulation run without any interactions, simply allowing the particles to orbit the sun. I then took their mean distance over a number of orbits and subtracted their starting radius, giving me the following result:

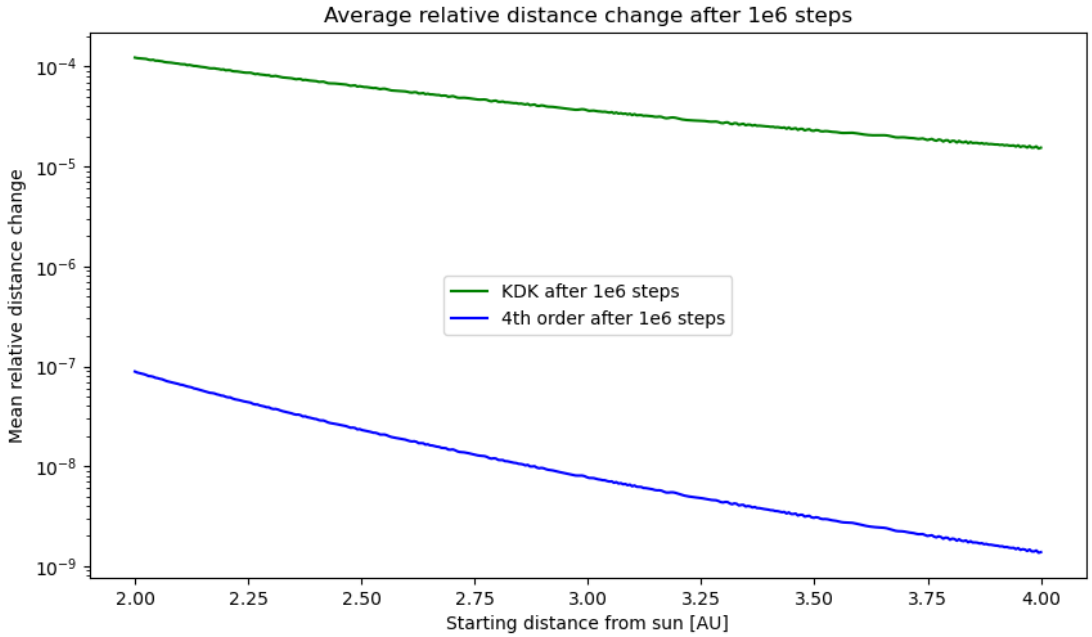


Figure 2: How good?

As can be seen, the improvement is immense!

That the inner planets are less precise makes sense, but you might ask: Why does the error seem to be slightly oscillating? Honestly I have no idea. I see two possible explanations: 1. It is the result of some numerical floating point stuff, or 2. I made a mistake in the implementation invalidating all of my results :)

6 Conclusion

For simulating the long term dynamics of an interacting set-up like our solar system, multiple different precautions need to be taken, as the equations needs to be optimized for this purpose. Therefore, smarter algorithms like high-order symplectic integration algorithms are needed.

When the correct systems are implemented, the Kirkwood gaps are clearly visible in simulation, at exactly the theoretically predicted radii.

7 Given more time and pages

When adding eccentricities to the orbits, the Kirkwood gaps were a lot less prominent. I had a hard time coming up with a way to quantify this (ie. making interesting plots akin to figure 2), so ultimately this had to be cut.

I would have loved to clean up my code for you as my testing mainly consisted of commenting/uncommenting lines of code making it unclear what happens when.

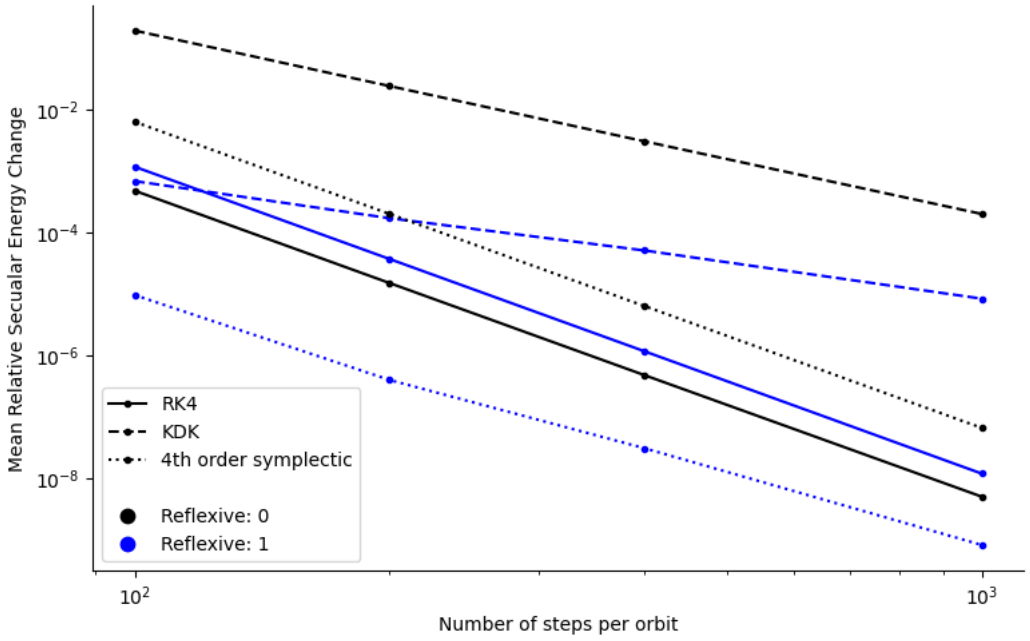
Finally, the main focus of this project ended up being the Kirkwood gaps and the computer science needed for the underlying simulation. I spent some time simulating the planets in our solar system, seeing how

the eight orbits (aka. an orbyte[5]) interacted at different parameter-changes, quantifying the stability of our solar system. These experiments were ultimately cut for a more clean project.

[5]A little Computational Astrophysics joke for you

A Secular Energy Change

Here is a comparison of secular/long-term error for fourth order accurate versions of the Yoshida (symplectic) and Runge-Kutta along with a single KDK as a base-line:



As suspected, the two symplectic integrators gain the most by having reflexive timesteps - especially for higher dt's (ie. a lower number of steps per orbit).

I came up with this measure for the week 7 assignment. I have run 50 bodies over at least ten orbits, and then fittet the resulting change in energy, hopefully getting an estimate for the long-term change this way instead of just the inter-orbit energy fluctuations. Two examples fits for the the fourth order symplectic integrator can be seen here: